UNIVERSITÉ DE MONTRÉAL

AJUSTEMENT DES VARIABLES DUALES DANS LE CONTEXTE D'UNE MÉTHODE DE GÉNÉRATION DE COLONNES

VIVIANE ROCHON
DÉPARTEMENT DE MATHÉTIQUES
ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
NOVEMBRE 1997

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

AJUSTEMENT DES VARIABLES DUALES DANS LE CONTEXTE D'UNE MÉTHODE DE GÉNÉRATION DE COLONNES

présenté par: ROCHON Viviane

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de:

Mme LAPIERRE Sophie, Ph.D., présidente

M. SOUMIS François, Ph.D., membre et directeur de recherche

M. DESAULNIERS Guy, Ph.D., membre et codirecteur de recherche

M. DESROSIERS Jacques, Ph.D., membre

À tous les étudiants

de l'École Polytechnique de Montréal.

Remerciements

Tout d'abord, je tiens à remercier ardemment Jacques Desrosiers, Guy Desaulniers et François Soumis pour leur aide inestimable dans la rédaction de ce mémoire.

Je remercie le FCAR (Fonds pour les Chercheurs et l'Aide à la Recherche), ainsi que François Soumis, pour leur soutien financier, sans lequel je n'aurais jamais fait cette maîtrise.

Merci à tout le personnel du GERAD, à tous les joueurs de tarot pour les heures de plaisir, à tous les étudiants pour leur chaleureuse complicité.

Je remercie tous mes amis et tous mes amours, sans lesquels il n'y aurait pas de raison de vivre.

Enfin, je voudrais remercier Gerolamo Cardano (Jérôme Cardan), mathématicien, philosophe et médecin italien (Pavie 1501-Rome 1576), pour avoir donné son nom à l'ordinateur hp9000/715 qui m'a beaucoup servi.

Résumé

Ce mémoire s'intéresse à l'optimisation de problèmes de routage par l'utilisation de la méthode de décomposition par génération de colonnes. Cette méthode décompose un problème donné en deux nouveaux problèmes: le problème maître et le sous-problème. Depuis 15 ans, la recherche effectuée au GERAD sur cette méthode de décomposition a surtout porté sur la résolution du sous-problème, laissant la résolution du problème maître à un optimiseur externe utilisant la méthode du simplexe. Ce mémoire innove en s'attaquant à la résolution du problème maître. L'objectif est de réduire les temps de résolution du problème maître, qui sont en général les plus importants pour les problèmes de grande taille. Plus particulièrement, nous nous intéressons à deux méthodes heuristiques pour résoudre le problème dual associé au problème maître. Ces deux méthodes, un **algorithme de sous-gradient** et un **algorithme de stabilisation des variables duales**, sont analysées, implantées et testées en détail. L'implantation de ces algorithmes s'intègre à un optimiseur nommé GENCOL, qui est développé au GERAD. Cet optimiseur est basé sur une approche de génération de colonnes.

L'algorithme de sous-gradient est basé sur celui proposé par Fisher et Kedia (1990) pour la résolution duale d'un problème de recouvrement / partitionnement de tâches. Cet algorithme tire profit de la relaxation de toutes les contraintes du problème primal dans la fonction objectif.

L'algorithme de stabilisation des variables duales a été développé par Du Merle, Villeneuve, Desrosiers et Hansen (1997). Cet algorithme part de l'hypothèse que si les variables duales peuvent être stabilisées pendant la résolution du problème maître, leur convergence sera plus rapide, et la résolution accélérée. Ceci vient de l'observation du comportement des valeurs des variables duales au cours de la résolution d'une relaxation linéaire: celles-ci oscillent beaucoup, de manière assez chaotique, avant de se stabiliser vers leurs valeurs définitives. L'algorithme de stabilisation parvient à borner les valeurs des variables duales en utilisant des variables primales de perturbation.

Des tests sur la relaxation linéaire de problèmes de m-voyageurs de commerce avec fenêtres de temps (m-TSPTW) ont montré que l'algorithme de sous-gradient permet de réduire les temps de résolution de 28% en moyenne. Quant à l'algorithme de stabilisation, les premiers résultats obtenus avec cette méthode sont mitigés. Il semble que le temps sauvé par une convergence plus rapide des valeurs des variables duales soit contrebalancé par une résolution plus lente du problème maître, due au surplus de variables primales. Curieusement, la combinaison des deux algorithmes donne une réduction du temps de résolution moyen de 29,7% par rapport au temps de résolution standard (c'est-à-dire avec l'algorithme du simplexe utilisé seul). Ceci est légèrement meilleur que lorsque l'algorithme de sous-gradient est utilisé seul. De plus, l'utilisation des deux algorithmes semble induire une convergence plus rapide de la valeur de l'objectif.

Ce mémoire a montré la pertinence d'utiliser des heuristiques duales dans le contexte de la méthode de génération de colonnes. Les algorithmes implantés dans le cadre de ce mémoire sont utilisables directement dans le cadre de l'optimiseur GENCOL.

Abstract

The subject of this thesis is the optimization of routing problems using the column generation method. This method breaks up a given problem into two new problems: a master problem and a subproblem. For 15 years, the research at GERAD has focused on the subproblem, leaving the master problem's resolution to an extern optimizer. This work innovates by considering the master problem. We want to accelerate the master problem's time, generally greater than the subproblem's in large problems. More precisely, we are interested in two heuristic methods for the dual problem associated to the master problem. These two methods, a **subgradient algorithm** and a **dual variable stabilization algorithm**, are analyzed, implemented and tested. These implementations are integrated to an optimizer, called GENCOL, developed at the GERAD, which uses the column generation algorithm.

The subgradient algorithm is based upon an algorithm proposed by Fisher and Kedia (1990), for the dual resolution of a set covering / set partitioning problem. This algorithm takes advantage of the relaxation of all primal constraints in the objective function.

The dual variable stabilization algorithm has been developped by Du Merle, Villeneuve, Desrosiers and Hansen (1997). This algorithm starts with the hypothesis that if the dual variables are stabilized during the master problem resolution, they converge faster, and the resolution is accelerated. This comes from the observation of

the dual variables' behavior within the resolution of a linear relaxation: they oscillate a lot, in quite a chaotic manner, before stabilizing around their definitive value. The stabilization algorithm manages to bound the dual variables by using supplementary primal variables, the perturbation variables.

The tests are made on the first linear relaxation of some problems. Tests on the subgradient algorithm show an average resolution time that is 28% faster. As for the stabilization algorithm, the first results obtained by this method are not very good. It seems that the time saved by a quicker convergence of the dual variables is lost by a slower resolution of the master problem, this being due to the addition of primal variables. Curiously, the combination of both algorithms reduces the average solution time by 29,7% which is better than when the subgradient algorithm is used alone.

This dissertation showed the pertinence of using dual heuristic methods within a column generation context. The algorithms implemented in the context of this research project are usable directly within the GENCOL optimizer.

Table des matières

Liste des tableaux

Liste des figures

Chapitre 1

Introduction

Ce chapitre constitue une introduction générale au problème traité dans ce mémoire. La section 1.1 introduit le genre de problèmes à résoudre dans le cadre de ce mémoire, ainsi que leurs difficultés. La section 1.2 donne une idée de la manière dont ces problèmes peuvent être résolus à l'aide d'une méthode de génération de colonnes (cette méthode sera décrite plus en détails dans le chapitre 2). L'objectif du mémoire, suivi du plan du mémoire, sont exposés dans la section 1.3.

1.1 Types de problèmes étudiés: les problèmes de routage

Les problèmes de routage sont des problèmes logistiques rencontrés couramment en optimisation. Il peut s'agir, par exemple, de planifier des itinéraires de camions dans un réseau routier ou de transporter de la marchandise de chez des fournisseurs vers des clients. Ce type de problèmes survient aussi en dehors des compagnies de transport comme, par exemple, dans les usines ou sur les chantiers: il s'agit alors de transporter de la marchandise entre les différentes parties de l'usine ou du chantier. Enfin, certains problèmes de confection d'horaires peuvent être modélisés comme des problèmes de routage sur un graphe d'états.

La résolution d'un problème de routage augmente avec le nombre de variables. Ce nombre est d'autant plus grand que les possibilités de solutions sont nombreuses. Ces possibilités se trouvent multipliées lorsqu'il s'agit d'établir une planification sur une semaine ou sur un mois type. D'autre part, les contraintes nombreuses (pouvant être induites par des règles de convention collective détaillées) contribuent elles aussi à compliquer le problème. Ces problèmes sont souvent trop complexes pour être efficacement résolus à la main. Même avec un ordinateur, ils deviennent rapidement trop gros pour être résolus avec les algorithmes standards: soit l'espace mémoire manque, soit le temps de calcul est trop long. C'est pourquoi de nouvelles méthodes sont constamment développées, pour permettre de résoudre de plus gros problèmes plus rapidement.

1.2 Les problèmes de routage résolus à l'aide d'une méthode de génération de colonnes

Une approche classique pour la résolution des problèmes de grande taille est la décomposition mathématique du problème initial en un problème maître et un sous-problème, à la manière de Dantzig et Wolfe (1961). Une structure de réseau sous-jacente aux problèmes de routage peut être mise en évidence. Dans cette structure, les tâches à couvrir sont associées à des noeuds ou à des arcs du réseau. Une solution d'un problème est alors représentée par un ensemble de chemins sur le réseau, tel que chacune des tâches soit couverte par un des chemins.

Lors de la résolution, deux procédures se partagent le travail. D'une part, le sous-problème détermine les chemins intéressants dans le réseau sous-jacent, c'est-à-dire

ceux dont le coût est avantageux par rapport aux chemins déjà trouvés. D'autre part, le problème maître construit une solution globale réalisable avec ces chemins. Le rôle du problème maître est également de calculer les variables duales qui serviront à générer de nouveaux chemins.

La méthode de décomposition de Dantzig-Wolfe est utilisée au GERAD[1] depuis 1981. Cette méthode, aussi connue sous le nom de méthode de génération de colonnes, consiste à construire par un processus itératif le problème maître en générant au besoin les colonnes, ou variables, de celui-ci. Elle est mise en pratique dans le logiciel GENCOL, avec des succès toujours grandissants. Beaucoup d'efforts ont été mis pour accélérer le processus de résolution de cette méthode. Jusqu'à maintenant, la plupart de ces efforts ont porté sur la résolution du sous-problème (algorithmes de programmation dynamique de plus court chemin avec contraintes, heuristiques de résolution approximative, stratégies de résolution). Pour le problème maître, un optimiseur externe utilisant l'algorithme du simplexe a toujours été utilisé. CPLEX est actuellement le troisième logiciel commercial utilisé, après les logiciels LANDP et XMP.

Le logiciel CPLEX utilise l'algorithme du simplexe (primal ou dual) pour résoudre un problème linéaire général. Il est très rapide pour de petits problèmes. Mais lorsque confronté à de gros problèmes (1000 tâches et plus), il est ralenti par la dégénérescence très fréquente de ces problèmes et les nombreuses réoptimisations à effectuer. Étant donné que le problème maître, essentiellement un problème de recouvrement ou de partitionnement de tâches, est plus simple qu'un problème linéaire général, l'utilisa-

[1]Le Groupe d'Études et de Recherche en Analyse des Décisions (GERAD) est un centre pluri-universitaire montréalais affilié à l'École Polytechnique, l'École des Hautes Études Commerciales, l'Université McGill et l'Université du Québec à Montréal.

tion de méthodes de résolution plus spécifiques au problème traité pourrait accélérer les temps de calcul. D'autre part, le problème maître de la méthode de génération de colonnes n'a pas besoin d'être résolu jusqu'à l'optimalité à chaque fois que des colonnes sont générées: des méthodes heuristiques pourront donc remplacer l'algorithme du simplexe pour certaines itérations de la méthode de génération de colonnes.

1.3 Objectifs du mémoire

Comme il a été vu à la section précédente, il semble intéressant d'accélérer la résolution du problème maître dans le cadre d'une méthode de génération de colonnes. Du même coup, si tout va bien, le temps total de résolution sera accéléré. Dans le cas des problèmes de routage, le problème maître est essentiellement un problème de recouvrement/partitionnement de tâches. L'approche utilisée sera donc basée sur des algorithmes spécifiques pour ce genre de problème.

Un rôle important du problème maître est de calculer les variables duales dirigeant, au niveau du sous-problème, la recherche de nouveaux chemins permettant d'améliorer la solution courante. Une façon de s'attaquer au problème maître est donc d'observer comment se comportent les variables duales tout au long de la résolution d'un problème par une méthode de génération de colonnes. Un exemple de ce comportement est illustré à la figure 1.1, où les valeurs prises par les variables duales sont données en fonction de l'itération de la méthode de génération de colonnes. Une telle itération est constituée par la résolution successive du sous-problème et du problème maître.

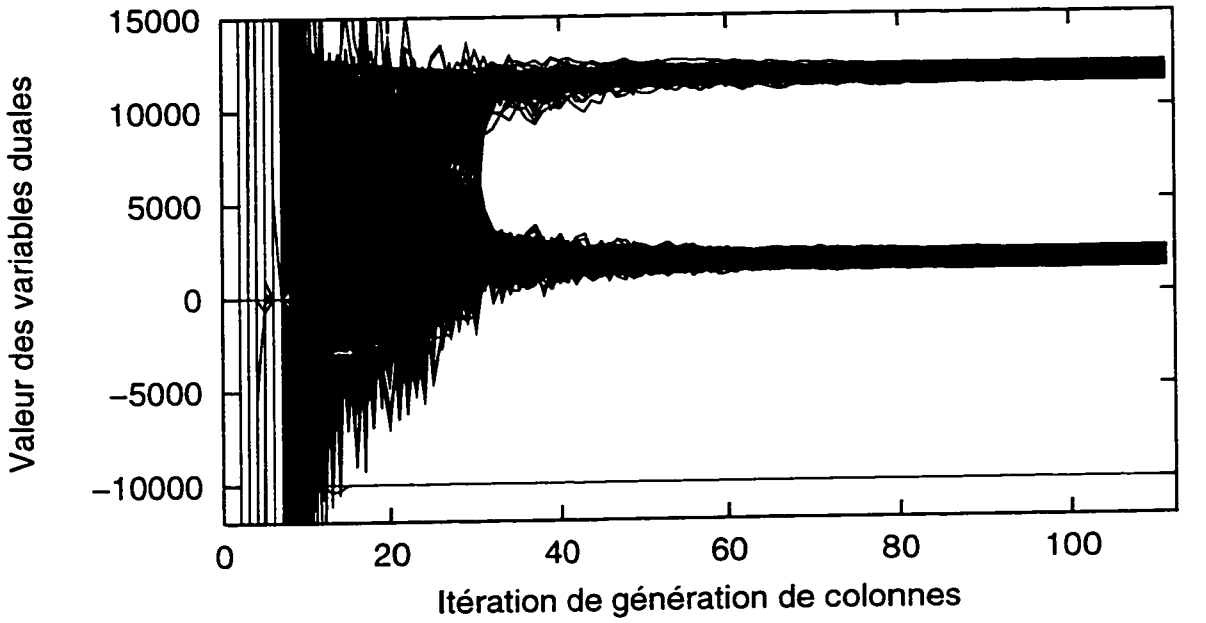


Figure 1.1: Comportement des variables duales pendant la résolution d'une relaxation linéaire par la méthode de génération de colonnes

Dans cette figure, il est possible d'observer que les valeurs des variables duales varient beaucoup avant de se stabiliser et de se rapprocher de leur valeur finale (valeur finale pour la résolution optimale de la relaxation linéaire du problème maître). Donc il pourrait être intéressant de tenter de lisser les courbes de ce graphe, c'est-à-dire de limiter l'amplitude du mouvement des variables d'une itération à l'autre. Les variables duales se rapprocheraient ainsi plus rapidement de leur valeur finale, ce qui pourrait réduire le nombre d'itérations de la méthode de génération de colonnes pour la résolution de la relaxation linéaire, et rendre ainsi la résolution plus rapide.

L'objectif de ce mémoire est donc de résoudre le problème maître avec de nouvelles méthodes qui seront intégrées à la méthode de génération de colonnes par l'entremise du logiciel GENCOL. Ces méthodes peuvent être heuristiques (pour plus de rapidité), en autant que le résultat final demeure exact. Nous nous attaquerons au dual du

problème en essayant de mieux contrôler les variables duales.

Le chapitre 2 présente quelques méthodes classiques mathématiques de décomposition utilisées pour des problèmes de routage semblables à ceux décrit dans la section 1.1. La relaxation lagrangienne (Geoffrion 1974) et la décomposition de Dantzig-Wolfe (1961) y sont décrites. Ce chapitre traite aussi de l'ajustement des variables duales par la méthode de sous-gradient. Suivent les caractéristiques de l'implantation de la méthode de décomposition de Dantzig-Wolfe (aussi connue sous le nom de méthode de génération de colonnes) dans le logiciel GENCOL. Enfin, les stratégies de résolution en nombres entiers sont abordées. Le chapitre 3 décrit plus spécifiquement les problèmes utilisés pour tester les algorithmes implantés dans le cadre de ce mémoire. Le chapitre 4 présente un algorithme de sous-gradient, permettant de résoudre de manière heuristique le problème maître dans le contexte d'une méthode de génération de colonnes. Les résultats de l'implantation de cet algorithme dans le logiciel GENCOL sont également présentés. Le chapitre 5 expose un second algorithme utilisant des stratégies de perturbation permettant de borner les variables duales du problème maître dans le contexte d'une méthode de génération de colonnes. La manière dont cet algorithme est implanté dans le logiciel GENCOL, et les résultats de cette implantation, sont présentés par la suite. Enfin, une courte conclusion fait le point sur les algorithmes étudiés et implantés dans le cadre de cette maîtrise, sur les résultats trouvés et sur les nouvelles directions de recherche intéressantes.

Chapitre 2

Méthodes de décomposition

Plusieurs méthodes de décomposition peuvent être employées pour résoudre un problème de grande taille. Les méthodes de décomposition de Dantzig-Wolfe (utilisant la programmation linéaire), de la relaxation lagrangienne (utilisant l'algorithme de sous-gradient), de faisceaux (utilisant la programmation quadratique) et du centre analytique (utilisant la méthode de points intérieurs) sont parmi les plus connues.

Ce chapitre décrit premièrement la méthode de la relaxation lagrangienne (section 2.1), puis la méthode de décomposition de Dantzig-Wolfe, aussi connue sous le nom de méthode de génération de colonnes (section 2.2). La section 2.3 décrit l'algorithme de sous-gradient qui sera intégré par la suite à la méthode de génération de colonnes. Les caractéristiques de l'implantation de la méthode de génération de colonnes dans l'optimiseur GENCOL sont données à la section 2.4. Enfin, la section 2.5 explique l'utilité d'un algorithme d'évaluation et de séparation progressive.

Dans les deux prochaines sections portant sur les méthodes de décomposition, nous considérons IP, un programme linéaire en nombres entiers, défini comme suit:

$$Z_{IP} = \min \sum_{j \in J} c_j X_j \tag{2.1}$$

$$\textit{sujet à} \quad \sum_{j \in J} a_{ij} X_j = b_i, \quad \forall i \in I \tag{2.2}$$

$$X \in \chi_E \tag{2.3}$$

où Z_{IP} dénote la valeur optimale de IP; $X = (X_j | j \in J)$ est le vecteur des variables de décision; $A = (A_{ij} | i \in I, j \in J)$ est la matrice des coefficients des contraintes; $b = (b_i | i \in I)$ est le vecteur des membres de droite; et χ_E est l'ensemble des points entiers d'une région convexe bornée χ définie par un ensemble de contraintes linéaires. Le problème obtenu par la relaxation linéaire des contraintes d'intégrité, et défini sur $\{X : X \in \chi\}$, est dénoté LP; sa valeur optimale est notée Z_{LP}.

2.1 La relaxation lagrangienne

La méthode de relaxation lagrangienne appliquée au problème (2.1)-(2.3) consiste à introduire les contraintes (2.2) dans l'objectif de IP et de profiter ainsi de la structure particulière de l'ensemble χ_E.

Associant les multiplicateurs $\pi = (\pi_i \in \mathbb{R} | i \in I)$ aux contraintes de (2.2) dans le problème IP, la **fonction lagrangienne** est définie comme suit:

$$L(X, \pi) = \sum_{j \in J} c_j X_j + \sum_{i \in I} \pi_i (b_i - \sum_{j \in J} a_{ij} X_j).$$

La minimisation de $L(X, \pi)$ sur l'ensemble χ_E donne la **fonction duale**:

$$\begin{aligned} L_D(\pi) &= \min_{X \in \chi_E} L(X, \pi) \\ &= \sum_{i \in I} b_i \pi_i + \min_{X \in \chi_E} \sum_{j \in J} (c_j - \sum_{i \in I} a_{ij} \pi_i) X_j. \end{aligned}$$

Il a été montré par Geoffrion (1974) que $L_D(\pi)$ constitue une borne inférieure à la valeur Z_{IP}, c'est-à-dire $L_D(\pi) \leq Z_{IP}$ pour tout vecteur π. La meilleure borne est donc donnée par la solution du **problème dual lagrangien**:

$$Z_{LR} = \max_{\pi \in \mathbb{R}} L_D(\pi) \leq Z_{IP}.$$

L'inégalité précédente est généralement stricte, mais le fait de conserver des contraintes d'intégrité dans χ_E permet de s'approcher de la valeur de Z_{IP}. Lorsque l'optimisation de la fonction lagrangienne sur χ_E donne le même résultat que sur χ, c'est-à-dire lorsque

$$\min_{X \in \chi_E} L(X, \pi) = \min_{X \in \chi} L(X, \pi),$$

le problème possède la **propriété d'intégrité**. Dès lors, la borne inférieure calculée par la relaxation lagrangienne ne donne pas une meilleure valeur que la relaxation linéaire de (2.1)-(2.3), i.e. $Z_{LR} = Z_{LP}$. D'où le résultat suivant:

$$Z_{LP} \leq Z_{LR} \leq Z_{IP}.$$

Si le gap $Z_{IP} - Z_{LR}$ est petit, le choix des contraintes à dualiser importe peu au niveau de la valeur de la borne: la relaxation qui facilite le calcul de $L_D(\pi)$ est donc choisie. Par contre, si $Z_{IP} - Z_{LR}$ est grand, le choix de l'ensemble χ_E formant les contraintes du problème lagrangien est important. L'utilisation adéquate des contraintes d'intégrité de χ_E permet souvent de déterminer une borne lagrangienne Z_{LR} très proche de Z_{IP}. Dans les deux cas et pour la plupart des applications, un arbre de branchement est nécessaire pour l'obtention de solutions entières.

La résolution du problème dual lagrangien passe par la détermination d'une suite de vecteurs des multiplicateurs π de façon à déterminer la meilleure borne inférieure.

Cette recherche (ou cet ajustement) des multiplicateurs est contrôlée par un problème maître. L'optimisation $\min_{X \in \chi_E} L(X, \pi)$ pour un vecteur de multiplicateurs π donné est appelée **sous-problème** et requiert généralement l'utilisation d'un algorithme profitant de la structure de χ_E.

L'optimisation du problème maître peut se faire de diverses façons. L'ajustement des multiplicateurs π peut se faire par programmation linéaire: c'est la méthode de Dantzig-Wolfe (1961), décrite à la section 2.2. La méthode la plus connue est certainement celle de sous-gradient popularisée par Held, Wolfe et Crowder (1974) pour le problème du voyageur de commerce. Cette méthode est décrite à la section 2.3. Enfin, des méthodes non linéaires peuvent être utilisées, telles la méthode des faisceaux utilisant la programmation quadratique (Lemaréchal 1989) ou encore la méthode ACCPM (Analytic Center Cutting Plane Method) des centres analytiques (Vial et Goffin 1992, Du Merle 1995) faisant appel à un algorithme de points intérieurs. Ces deux dernières méthodes ne sont pas décrites dans ce mémoire.

2.2 La décomposition de Dantzig-Wolfe

Le processus de décomposition de Dantzig-Wolfe peut être décrit de la façon suivante. Tout point de l'intersection des contraintes (2.2)-(2.3) peut s'exprimer comme combinaison convexe des points de l'enveloppe convexe de χ_E, soit $Conv(\chi_E)$. Ici l'hypothèse non restrictive en pratique sera faite que χ_E est un ensemble borné. Notons par $X_p = (X_{jp} | j \in J)$ chacun des points extrêmes de $Conv(\chi_E)$ et par Ω l'ensemble de ces points extrêmes. Tout point $X \in \chi$ satisfaisant (2.2) peut donc s'exprimer par

$$X = \sum_{p \in \Omega} \theta_p X_p$$

$$\begin{aligned} \sum_{p\in\Omega} \theta_p &= 1 \\ \theta_p &\geq 0,\ \forall p \in \Omega. \end{aligned}$$

En faisant la substitution dans (2.1) et (2.2), le problème maître suivant est obtenu:

$$\begin{aligned} & \min \sum_{j\in J} c_j \Big(\sum_{p\in\Omega} \theta_p X_{jp}\Big) \\ \textit{sujet à} \quad & \sum_{j\in J} a_{ij}\Big(\sum_{p\in\Omega} \theta_p X_{jp}\Big) = b_i,\ \forall i \in I \\ & \sum_{p\in\Omega} \theta_p = 1 \\ & \theta_p \geq 0,\ \forall p \in \Omega \end{aligned}$$

où les poids $\theta_p, p \in \Omega$, associés aux points extrêmes dans la combinaison convexe correspondent aux nouvelles variables de décision.

Après réorganisation des sommations, le problème maître se présente sous la forme:

$$\begin{aligned} & \min \sum_{p\in\Omega} \Big(\sum_{j\in J} c_j X_{jp}\Big)\theta_p \\ \textit{sujet à} \quad & \sum_{p\in\Omega} \Big(\sum_{j\in J} a_{ij} X_{jp}\Big)\theta_p = b_i,\ \forall i \in I \\ & \sum_{p\in\Omega} \theta_p = 1 \\ & \theta_p \geq 0,\ \forall p \in \Omega. \end{aligned}$$

L'ajustement du vecteur π des multiplicateurs $\pi_i \in \mathbb{R},\ \forall i \in I$ associés aux contraintes indicées par i, ainsi que du multiplicateur $\lambda \in \mathbb{R}$ associé à la contrainte de combinaison convexe, passe par la résolution du programme linéaire précédent. Ces multiplicateurs (π, λ), vont servir à déterminer si un nouveau point extrême $X_p, p \in \Omega$ peut améliorer la solution courante, i.e., s'il existe une variable $\theta_p, p \in \Omega$ de coût marginal négatif. Pour ce faire, le sous-problème suivant doit être résolu:

$$\min_{p\in\Omega} \sum_{j\in J} c_j X_{jp} - \sum_{i\in I} \pi_i \Big(\sum_{j\in J} a_{ij} X_{jp}\Big) - \lambda$$

qui s'écrit également comme

$$-\lambda + \min_{X \in X_E} \sum_{j \in J} (c_j - \sum_{i \in I} a_{ij}\pi_i) X_j.$$

Ce sous-problème est, à une constante près, le sous-problème lagrangien. Il permet d'une part de calculer une borne inférieure sur la valeur de Z_{IP} et, d'autre part, de cesser le processus de génération si le résultat de l'optimisation précédente ne donne pas de variables de coût marginal négatif.

Ainsi, l'ajustement des multiplicateurs π se fait de manière équivalente, qu'il s'agisse du problème dual lagrangien dans la méthode de la relaxation lagrangienne, ou d'un problème de programmation linéaire dans la méthode de Dantzig-Wolfe. Ces deux méthodes sont équivalentes.

2.3 L'algorithme de sous-gradient

La fonction duale $L_D(\pi)$ vue dans la section 2.1 est continue, concave et linéaire par morceaux. Elle n'est pas dérivable partout, ce qui la rend difficile à maximiser. Pour tout vecteur de direction d donné, elle a comme dérivée directionnelle

$$L'_D(\pi; d) = \lim_{t \to 0^+} \frac{L_D(\pi + td) - L_D(\pi)}{t}.$$

Si $L_D(\pi)$ était dérivable partout, un extremum se trouverait en un point où la différentielle $\frac{\partial L_D(\pi)}{\partial \pi} = 0$. Mais, dans le cas présent, le maximum risque d'être en un point singulier où il existe plusieurs dérivées directionnelles valides. Heureusement, la notion de différentiabilité d'une fonction est étendue pour nous donner ce dont nous avons besoin.

Le sous-gradient est une extension de la notion de dérivée pour les fonctions non différentiables. Le vecteur u de dimension n, où n est la dimension de l'ensemble des contraintes (2.2), est un **sous-gradient** en $\pi \in \mathbb{R}^n$ de la fonction concave $L_D(\pi)$ définie sur $\mathbb{R}^n$ si

$$L_D(y) - L_D(\pi) \leq u \cdot (y - \pi)$$

pour tout $y \in \mathbb{R}^n$. L'ensemble de tous les sous-gradients en π est l'ensemble compact et convexe $\partial L_D(\pi)$, appelé le **sous-différentiel** en π. Dans le cas où le sous-différentiel en π est réduit à un seul élément, la fonction $L_D(\pi)$ est dérivable et $\partial L_D(\pi)$ est sa dérivée; de manière plus générale, $L'_D(\pi; d) \in \partial L_D(\pi)$, $\forall d$. Une condition nécessaire et suffisante pour que π maximise $L_D(\pi)$ est que $L'_D(\pi; d) \leq 0$, $\forall d$, ou de manière équivalente, que $0 \in \partial L_D(\pi)$.

Un sous-gradient en un point donné indique donc une bonne direction à suivre pour atteindre le maximum d'une fonction. C'est cette propriété qui est utilisée par l'algorithme de sous-gradient pour trouver le maximum de $L_D(\pi)$. L'idée de cet algorithme est de suivre itérativement les directions données par un sous-gradient jusqu'à être assez proche du maximum de $L_D(\pi)$.

Le vecteur de départ π^0 est souvent donné par un algorithme glouton (Fisher et Kedia 1990, Toth 1996). Intuitivement, la **direction** d sera choisie de manière à ce que $L'_D(\pi, d) > 0$, de telle sorte qu'en avançant d'un pas assez petit θ (c'est-à-dire en changeant π à $\pi + \theta d$) la valeur de $L_D(\pi)$ augmente, se rapprochant donc de son maximum. Un sous-gradient en π se calcule par

$$\max_{X \in X_B} \sum_{i \in I} b_i - \sum_{i \in I} \sum_{j \in J} a_{ij} X_j.$$

Cependant, par optimisme, cette direction sera choisie même si $L'_D(\pi, u) < 0$ (Held, Wolfe et Crowder, 1974).

Pour calculer le **pas** θ, l'idéal serait de calculer la valeur maximale que prend $L_D(\pi)$ dans la direction $\frac{\partial L_D(\pi)}{\partial \pi}$, mais ce calcul est très long. Une procédure heuristique de Held, Wolfe et Crowder (1974) est préférée afin de réduire le temps de calcul. Le pas dépend avant tout, de façon proportionnelle, de l'éloignement de π avec la valeur optimale π^* qui maximise $L_D(\pi)$: plus π est loin de π^*, plus le pas devra être long. Évidemment, la valeur de π^* n'est pas connue et c'est pourquoi une borne supérieure $\overline{Z}_{LR}$ sur $L_D(\pi^*)$ est nécessaire. Plus $\overline{Z}_{LR}$ est proche de $L_D(\pi^*)$, plus l'algorithme de sous-gradient sera performant. Dans l'approche classique utilisée dans ce mémoire, le pas est aussi fonction de l'inverse du carré de la norme de la direction.

Enfin, le pas contient un facteur multiplicateur ρ^k. Cette suite de paramètres $\{\rho^k\}$ (k indiquant l'indice de l'itération de l'algorithme de sous-gradient) assure la convergence de l'algorithme vers π^*, si $\lim_{k\to\infty} \rho^k = 0$ et $\sum_{k=0}^{\infty} \rho^k = \infty$. La série $\sum_{k=0}^{\infty} \rho^k$ peut aussi être convergente. Dans ce cas, la résolution est plus rapide que si la série $\{\rho^k\}$ diverge. La suite des vecteurs $\{\pi^k\}$ pourrait alors converger vers une autre valeur que la valeur optimale π^*, mais les résultats sont quand même très bons, comme dans Toth (1996), et dans Held, Wolfe et Crowder (1974), où une convergence de type géométrique est présente.

Ainsi, à partir d'une valeur duale π^0 donnée, un sous-gradient u^0 proposant une direction de déplacement intéressante est calculé, de même qu'un pas θ^0 correspondant à la distance à faire dans cette direction. Cela donne un vecteur de valeurs duales π^1 avec lequel sont recommencés les calculs d'un sous-gradient u^1 et d'un pas θ^1. Le processus se poursuit jusqu'à ce qu'un critère d'arrêt soit vérifié.

Il y a plusieurs critères d'arrêt pour l'algorithme de sous-gradient:

1. la valeur de la fonction duale, $L_D(\pi^k)$ à l'itération k a atteint la valeur de la borne supérieure $\overline{Z}_{LR}$, au gap admissible près choisi au départ;

2. le nombre d'itérations de l'algorithme de sous-gradient a atteint un maximum donné;

3. le paramètre ρ^k à l'itération k est devenu tellement petit que le pas θ^k est pratiquement nul et qu'il n'est plus intéressant de poursuivre la résolution par l'algorithme.

2.4 L'algorithme de Dantzig-Wolfe dans le logiciel GENCOL

Le logiciel GENCOL est une implantation informatique de l'algorithme de décomposition de Dantzig-Wolfe, aussi connu sous le nom de méthode de génération de colonnes. Dans ce logiciel, il y a alternance entre la résolution du problème maître et celle du sous-problème (voir figure 2.1). Le problème maître comprend l'objectif ainsi que les contraintes liantes. Les contraintes liantes sont celles qui portent sur plus d'un chemin à la fois. Dans le contexte étudié, le sous-problème comprend les contraintes de flot et toutes les autres contraintes internes à la définition d'un chemin. Le sous-problème se modélise en problème de plus court chemin avec variables de ressources dans le graphe sous-jacent au problème originel et se résout par programmation dynamique. La résolution du sous-problème donne un ensemble de chemins réalisables, et les variables de flot associées aux chemins ayant un coût marginal négatif (donc plus intéressants que les chemins trouvés précédemment) sont ajoutées au problème maître. Les chemins sont représentés par des colonnes de la matrice du problème

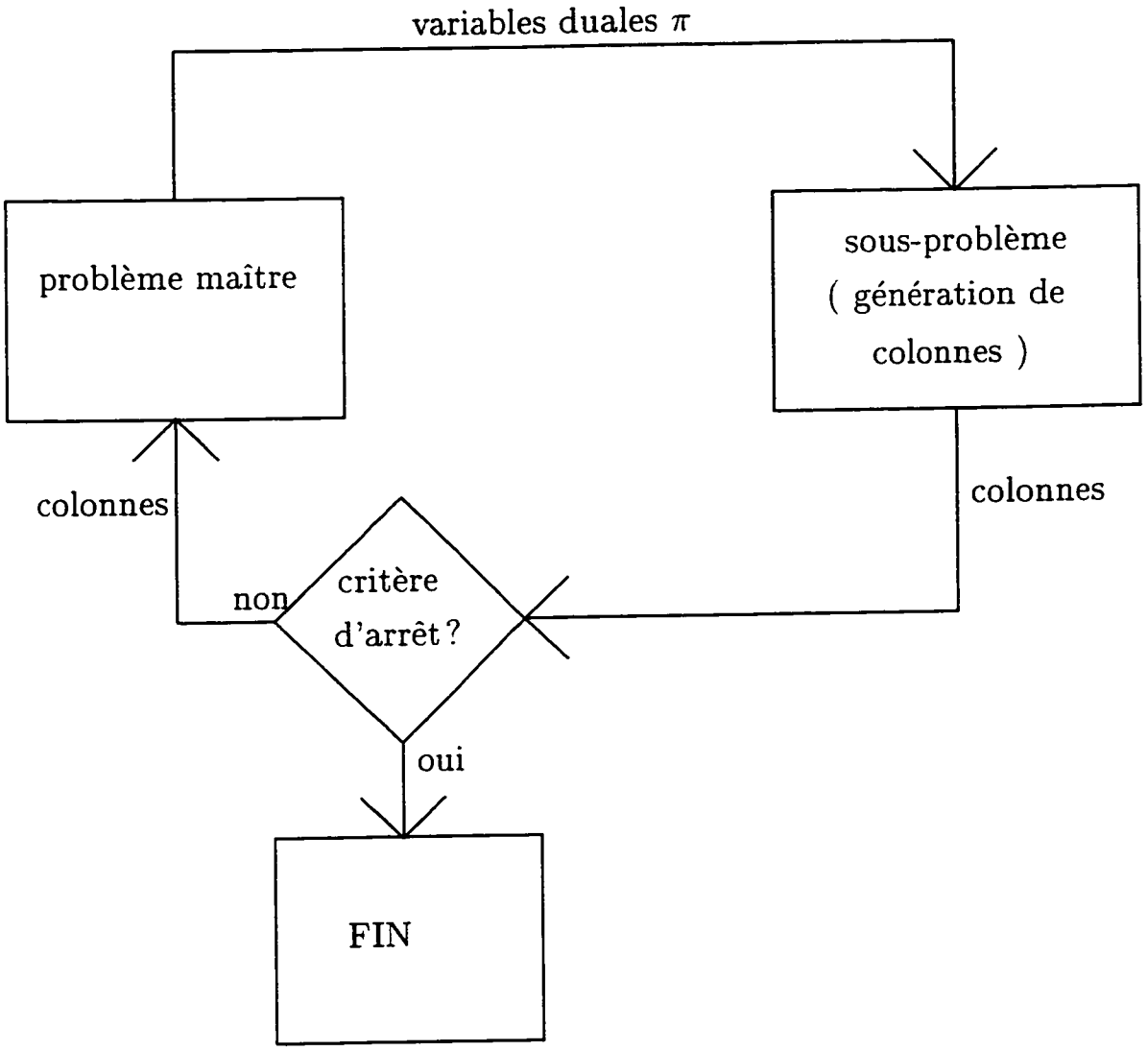


Figure 2.1: Processus itératif de résolution par une approche de génération de colonnes

maître, d'où le nom de "génération de colonnes".

Le problème maître trouve une combinaison des chemins retenus qui satisfait les contraintes liantes à coût minimum. Un chemin n'est pas forcément choisi un nombre entier de fois: par exemple, une tâche devant être couverte une fois peut être couverte à moitié par un chemin et à moitié par un autre chemin. Le problème maître est un problème de programmation linéaire qui est résolu, dans la méthode de génération

de colonnes et plus particulièrement dans l'optimiseur GENCOL, par l'algorithme du simplexe. Il est représenté par une matrice dont chaque colonne représente un chemin possible, et chaque variable le nombre de fois que le chemin correspondant est utilisé. Dans une approche duale comme la relaxation lagrangienne, le problème maître peut être résolu par un algorithme dual telle la méthode de sous-gradient (vue à la section 2.3).

Les valeurs duales π trouvées à l'optimalité du problème maître sont envoyées au sous-problème pour modifier les coûts. À chaque tâche i (du graphe sous-jacent au sous-problème) correspond une variable duale π_i. Le coût c_{ij} d'un arc (i,j) supportant la tâche i est modifié en posant $\hat{c}_{ij} := c_{ij} - \pi_i$. Plus la variable duale π_i est grande, plus la tâche i est attrayante, et moins il coûte cher d'utiliser l'arc (i,j).

Le sous-problème est de nouveau résolu, en tenant compte des nouveaux coûts, et la boucle problème maître - sous-problème est répétée jusqu'à ce qu'un critère d'arrêt soit atteint. Étant donné que le nouveau problème maître est le même que celui de l'itération précédente, avec un plus grand choix de variables grâce aux nouveaux chemins générés, la valeur de l'objectif obtenu par chaque résolution d'un nouveau problème maître s'améliore à chaque itération. Pour une résolution optimale du problème maître, le processus itératif s'arrête lorsqu'aucune nouvelle colonne ne peut être générée. Autrement, un critère d'arrêt heuristique peut être employé: par exemple lorsqu'une solution duale de valeur assez proche d'une borne supérieure prédéterminée est trouvée.

À ce moment, si la dernière solution obtenue est exacte et entière, c'est une solution de valeur optimale Z_{IP}. Sinon, elle permet d'obtenir une borne inférieure sur la valeur

de la solution optimale qui sera déterminée par un algorithme d'évaluation et de séparation progressive.

2.5 Recherche d'une solution entière

Les méthodes de la relaxation lagrangienne et de Dantzig-Wolfe, vues dans les sections 2.1 et 2.2, résolvent des problèmes linéaires. Or, bien souvent, le problème d'optimisation considéré est en nombres entiers, donc bien plus difficile à résoudre. Dans ce cas, des décisions de branchement, qui diffèrent selon le contexte, doivent être prises dans le cadre d'un algorithme d'évaluation et de séparation progressive.

Dans le cas de la relaxation lagrangienne, malgré que la résolution soit duale, le branchement se fait sur les variables primales. Une solution primale réalisable est donc nécessaire. En général, celle-ci est déterminée par une heuristique primale. La valeur de la solution duale sert alors à limiter le nombre de noeuds explorés.

Dans le cas de la décomposition de Dantzig-Wolfe, une solution primale possiblement fractionnaire est disponible à la fin de la résolution de la relaxation linéaire. La valeur de cette solution procure une borne inférieure à la solution optimale entière. Les décisions de branchement sont souvent prises sur la valeur des flots entre les tâches. De telles décisions sont imposées au niveau du sous-problème. Le branchement ne peut se faire sur les variables du problème maître étant donné que les variables violant les décisions prises peuvent toujours être générées de nouveau, à l'aide du sous-problème.

Chapitre 3

Problèmes tests

Les problèmes de routage ont été présentés de façon générale à la section 1.1. Dans ce chapitre, un type précis de problème de routage est décrit: le problème des m-voyageurs de commerce avec fenêtres de temps, dénoté **m-TSPTW** (multi-Traveling Salesman Problem with Time Windows). Ce type de problème consiste à déterminer les itinéraires d'un groupe de véhicules devant visiter à coût minimum un certain nombre de clients. Chaque client possède une fenêtre de temps déterminée pendant laquelle la visite doit avoir lieu.

Tous les tests effectués sur les implantations informatiques dans le cadre de ce mémoire seront faits sur ce type de problèmes. Deux raisons motivent ce choix. La première est que ce type de problèmes est suffisamment représentatif de la classe des problèmes de routage de véhicules et d'horaires d'équipages, sans faire intervenir toute la complexité des conventions de travail. La seconde provient du fait que, dans une méthode de décomposition comme celle de Dantzig et Wolfe, la majeure partie du temps de résolution pour ce type de problèmes se situe au niveau du problème maître. Le but du mémoire étant de réduire les temps de calcul du problème maître, les problèmes de type m-TSPTW constituent donc des candidats de choix.

La section 3.1 décrit en détails le m-TSPTW et introduit la notation utilisée dans

la section suivante, soit la section 3.2 qui présente un modèle mathématique pour le m-TSPTW. Enfin, la section 3.3 expose la formulation du problème maître obtenue en décomposant ce modèle mathématique selon le principe de Dantzig et Wolfe (1961).

3.1 Un problème de routage: le m-TSPTW

De manière générale, le modèle du m-TSPTW s'applique à des problèmes où un ensemble de tâches doivent être effectuées une fois chacune. De plus, chaque tâche doit être effectuée dans un intervalle de temps précis. Plusieurs ressources identiques sont disponibles pour effectuer les tâches.

Pour donner une idée plus concrète du problème à résoudre, considérons que les tâches à effectuer sont des clients à visiter. Chaque client doit être visité à l'intérieur d'un intervalle de temps précis appelé **fenêtre de temps**. Pour ce faire, des véhicules tous identiques sont disponibles, regroupés dans des dépôts. L'utilisation d'un véhicule entraîne un coût fixe prédéterminé. L'objectif du problème consiste à minimiser une somme pondérée du coût total de la solution et du nombre de véhicules utilisés.

Ce problème de multi-flots avec fenêtres de temps se modélise comme suit. L'ensemble des clients à visiter est noté N et les $|N|$ clients sont associés à $|N|$ noeuds-tâches. Par abus de notation, cet ensemble de noeuds est aussi noté N. L'ensemble des dépôts est noté K. Un dépôt $k \in K$ contient v^k véhicules. Il est représenté par une paire de noeuds: un dépôt-source $o(k)$ et un dépôt-puits $d(k)$. $N^k = N \cup \{o(k), d(k)\}$ est donc l'ensemble des noeuds relatifs au dépôt k.

La fenêtre de temps du client $i \in N$ est notée par l'intervalle $[a_i, b_i]$. Un véhicule

peut arriver chez un client $i \in N$ avant le temps a_i et attendre jusqu'à a_i sans coûts additionnels; par contre, il doit absolument arriver avant le temps b_i. De manière similaire pour les dépôts, un véhicule du dépôt k doit partir de ce dépôt entre les temps $a_{o(k)}$ et $b_{o(k)}$, et y revenir entre $a_{d(k)}$ et $b_{d(k)}$. Comme c'est souvent le cas en pratique, l'heure de départ du dépôt est fixée, i.e. $a_{o(k)} = b_{o(k)}\ \forall k \in K$.

Le temps de parcours du client $i \in N$ au client $j \in N$ est noté t_{ij} et comprend la durée de la visite chez le client i. De même, le temps de parcours du dépôt $k \in K$ au client $i \in N$ est noté $t_{o(k)i}$, et le temps de parcours du client $i \in N$ au dépôt $k \in K$ est noté $t_{id(k)}$, et comprend la durée de la visite chez le client i. Un véhicule provenant du dépot-source $o(k)$ peut aller chez le client $i \in N$ si le temps le permet, c'est-à-dire si $a_{o(k)} + t_{o(k)i} \leq b_i$. De même, après avoir visité le client i, le véhicule peut servir le client j si $a_i + t_{ij} \leq b_j$, ou revenir au dépôt-puits $d(k)$ si $a_i + t_{id(k)} \leq b_{d(k)}$. Il y a donc pour chaque dépôt $k \in K$, un sous-ensemble A^k d'arcs, avec $A^k \subseteq N^k \times N^k$. L'ensemble de tous les arcs est noté A (i.e. $A = \bigcup_{k \in K} A^k$). Notons que les fenêtres de temps ne compliquent pas nécessairement le problème. Elles peuvent même le simplifier, car elles limitent le nombre total d'arcs et réduisent donc le nombre de chemins à comparer.

Le coût c_{ij} d'un arc (i, j) est le coût réel de déplacement du client $i \in N$ au client $j \in N$ pour un véhicule quelconque. Les coûts $c_{o(k)i}$ des arcs dépôt-tâche et $c_{id(k)}$ des arcs tâche-dépôt sont définis de façon similaire. De plus, les coûts $c_{o(k)i}$ comprennent le coût fixe d'utilisation d'un véhicule. Pour minimiser le nombre de véhicules utilisés, il suffit de rendre ce coût fixe très grand.

Les variables du problème sont de deux types. Une variable binaire X_{ij}^k est associée

à chaque arc $(i,j) \in A^k, k \in K$. Elle indique le flot sur l'arc $(i,j) \in A^k$. Cette variable est égale à 1 si l'arc (i,j) est choisi dans la solution pour le dépôt k, à 0 sinon. Une variable de temps T_i^k est associée à chaque noeud $i \in N^k, k \in N$. Si $i \in N$, elle spécifie l'heure effective de début de service chez le client $i \in N$ d'un véhicule provenant du dépôt $k \in K$; sinon elle indique l'heure du premier départ du dépôt k $(i = o(k))$ ou l'heure de la dernière arrivée à ce dépôt $(i = d(k))$.

3.2 Un modèle mathématique pour le m-TSPTW

Étant donné la notation définie à la section précédente, le m-TSPTW peut s'écrire comme le problème de multi-flots avec fenêtres de temps suivant:

$$\min \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij} X_{ij}^k \tag{3.1}$$

sous les contraintes:

$$\sum_{k \in K} \sum_{j \in N^k} X_{ij}^k = 1, \quad \forall i \in N \tag{3.2}$$

$$\sum_{j \in N^k} X_{o(k)j}^k = \sum_{i \in N^k} X_{id(k)}^k \leq v^k, \quad \forall k \in K \tag{3.3}$$

$$\sum_{j \in N^k} X_{ji}^k - \sum_{j \in N^k} X_{ij}^k = 0, \quad \forall k \in K, \forall i \in N \tag{3.4}$$

$$X_{ij}^k (T_i^k + t_{ij} - T_j^k) \leq 0, \quad \forall k \in K, \forall (i,j) \in A^k \tag{3.5}$$

$$a_i \leq T_i^k \leq b_i, \quad \forall k \in K, \forall i \in N^k \tag{3.6}$$

$$X_{ij}^k \in \{0,1\}, \quad \forall k \in K, \forall (i,j) \in A^k. \tag{3.7}$$

- L'objectif (3.1) est de minimiser le coût total, incluant les coûts fixes des véhicules utilisés.

- Les contraintes (3.2) spécifient que chaque tâche doit être couverte exactement une fois, par un véhicule provenant de l'un des $|K|$ dépôts. Il est à noter que ces contraintes de partitionnement ($= 1$) peuvent être transformées, sans perte de réalisabilité ni d'optimalité, en contraintes de recouvrement (≥ 1) (i.e. chaque tâche doit être couverte au moins une fois) si l'inégalité du triangle est satisfaite pour les coûts et les temps de parcours associés aux arcs. En effet, si la tâche n est couverte deux fois dans la version avec contraintes de recouvrement, alors un des deux chemins la recouvrant contient disons les arcs (i, n) et (n, j); or, par l'inégalité du triangle, $c_{ij} \leq c_{in} + c_{nj}$ et $t_{ij} \leq t_{in} + t_{nj}$, et il n'est donc pas avantageux de couvrir n plus d'une fois.

- Les contraintes (3.3) assurent qu'il y ait autant de véhicules partant du dépôt que de véhicules y revenant et limitent le nombre de véhicules utilisés à chaque dépôt $k \in K$ au nombre de véhicule disponibles v^k.

- Les contraintes (3.4) obligent la conservation du flot en tout noeud i de N: tout le flot entrant en un de ces noeuds doit en sortir, i.e. tous les véhicules qui arrivent chez un client doivent en repartir.

- Les contraintes (3.5) assurent qu'il y ait suffisamment de temps disponible pour effectuer le déplacement entre les noeuds i et j et, s'il y a lieu (i.e. si $i \in N$), la tâche au noeud i. Par conséquent, une telle contrainte indique que, si l'arc $(i, j) \in A^k$ fait partie de la solution pour un véhicule provenant du dépôt $k \in K$ (i.e. si $X_{ij}^k > 0$), alors le temps de début de service au noeud j (ou d'arrivée au dépôt k si $j = d(k)$) doit être supérieur ou égal au temps de début de service au noeud i (ou de départ du dépôt k si $i = o(k)$) plus le temps de parcours t_{ij} entre ces deux noeuds.

- Les contraintes (3.6) représentent les fenêtres de temps pour chaque noeud.
- Finalement, les contraintes (3.7) sont les contraintes d'intégrité des variables de flot.

3.3 Formulation du problème maître

Comme illustré à la figure 3.1, la matrice des coefficients des contraintes du modèle proposé pour le m-TSPTW (3.1)-(3.7) a une structure angulaire de blocs. En fait, sans la présence des contraintes liantes (3.2) définissant un partitionnement des tâches, il y aurait $|K|$ petits problèmes à résoudre, un par dépôt. Cette structure particulière incite à utiliser la décomposition de Dantzig-Wolfe ou la relaxation lagrangienne (méthodes vues dans les sections 2.1 et 2.2) pour résoudre le problème. Pour chacun des dépôts $k \in K$, un sous-problème est défini comme un problème de plus court chemin avec fenêtres de temps. Le problème maître consiste à sélectionner parmi les solutions trouvées par les sous-problèmes, celles qui mènent à une solution optimale globale.

Soit P^k l'ensemble des chemins réalisables à partir du dépôt $k \in K$. Dénotons par c_p le coût du chemin $p \in P^k$, et par a_{ip}, un paramètre binaire qui prend la valeur 1 lorsque le chemin $p \in P^k$ visite le client $i \in N$, et 0 sinon. Finalement, définissons, pour chaque dépôt $k \in K$ et chaque chemin $p \in P^k$, une variable θ_p^k qui indique le flot de véhicules provenant du dépôt k sur le chemin p. Ces variables de chemin peuvent être restreintes à des valeurs binaires puisque chaque chemin contient au moins une visite chez un client et qu'il n'est pas avantageux de visiter deux fois le même client.

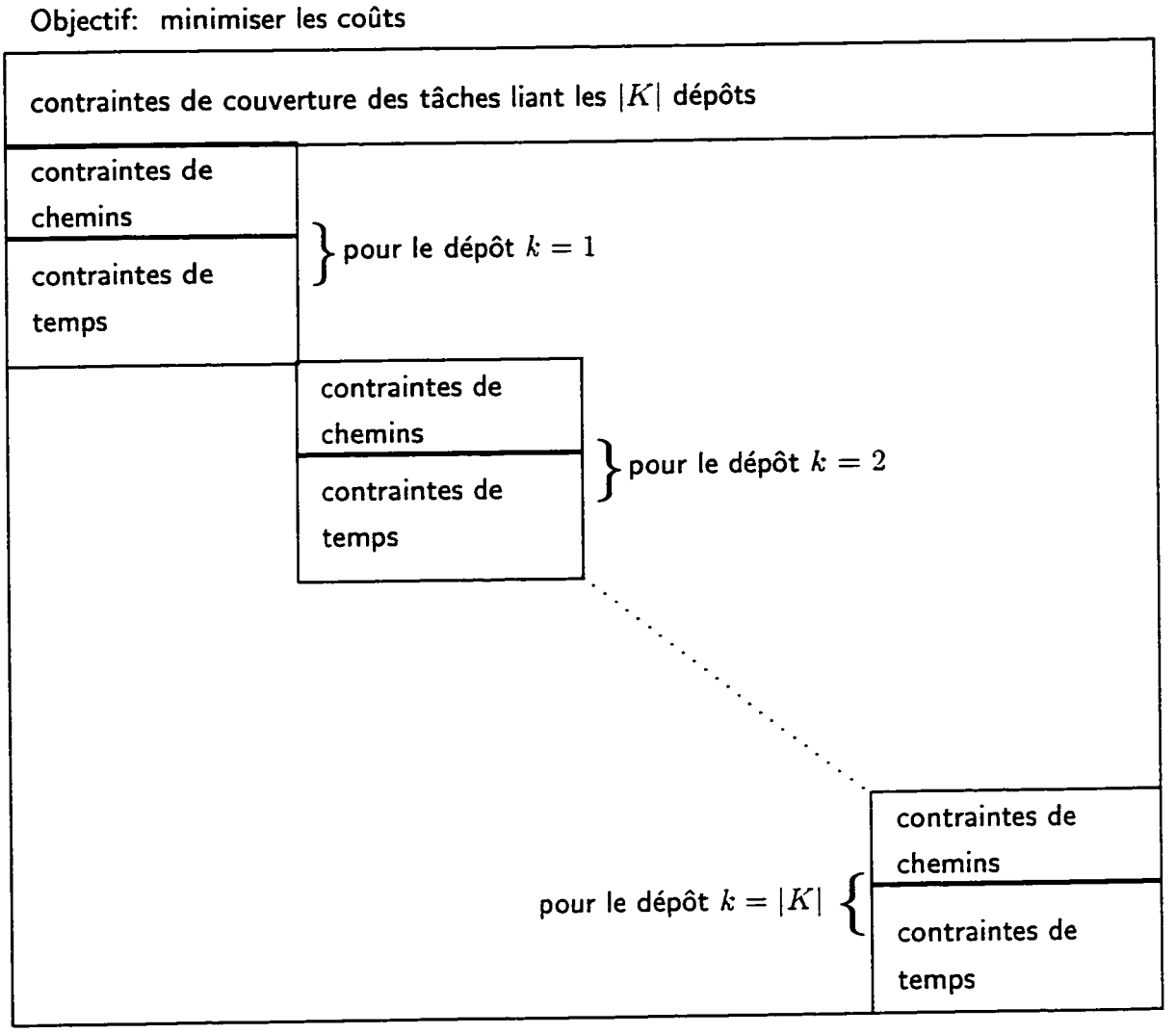


Figure 3.1: Structure de blocs de la matrice des coefficients

En utilisant cette notation, le problème maître peut se formuler ainsi:

$$\min \sum_{k \in K} \sum_{p \in P^k} c_p \theta_p^k \tag{3.8}$$

sous les contraintes

$$\sum_{k \in K} \sum_{p \in P^k} a_{ip} \theta_p^k = 1 \qquad \forall i \in I \tag{3.9}$$

$$\sum_{p \in P^k} \theta_p^k \leq Y^k \qquad \forall k \in K \tag{3.10}$$

$$\theta_p^k \in \{0,1\} \qquad \forall k \in K,\ \forall p \in P^k. \tag{3.11}$$

- L'objectif (3.8) consiste à minimiser les coûts totaux.
- Les contraintes (3.9) spécifient que chaque client doit être visité exactement une fois.
- Les contraintes (3.10) limitent le nombre de véhicules disponibles à chaque dépôt.
- Finalement, les contraintes (3.11) indiquent que les variables de chemin doivent prendre des valeurs binaires.

Il s'agit donc essentiellement d'un problème de partitionnement, avec pour contraintes supplémentaires liantes les limites sur le nombre de véhicules par dépôt.

Le fait que ce problème en soit principalement un de partitionnement, ou de recouvrement selon la formulation, invite à utiliser un algorithme spécialisé dans la résolution de ce type de problèmes, comme cela sera fait à la section 4.1. Dans ce cas, les contraintes liantes qui font que le problème n'est pas tout à fait un problème de partitionnement pourront être traitées à part. Cette manière de procéder est efficace dans un cas comme celui-ci où l'effet des contraintes supplémentaires est limité.

Chapitre 4

Intégration d'un algorithme de sous-gradient à l'algorithme de génération de colonnes

Le premier algorithme intégré à l'algorithme de génération de colonnes et pouvant remplacer l'algorithme du simplexe pour la résolution du problème maître est un algorithme de sous-gradient inspiré de l'algorithme de Fisher et Kedia (1990). L'algorithme de Fisher et Kedia est décrit dans la section 4.1. La section 4.2 explique pourquoi seul l'algorithme de sous-gradient a été retenu de l'algorithme de Fisher et Kedia, et comment il a été implanté dans le logiciel GENCOL, qui utilise la méthode de génération de colonnes. Enfin la section 4.3 donne les résultats numériques des tests effectués pour cette implantation.

4.1 L'algorithme de Fisher et Kedia

Cette section décrit l'algorithme proposé par Fisher et Kedia (1990). Tout d'abord, le contexte dans lequel cet algorithme s'applique est présenté. Ensuite, deux procédures heuristiques de l'algorithme de Fisher et Kedia, soit une procédure de type glouton et une procédure de type 3-opt, sont décrites en détail. Enfin, les étapes finales de

l'algorithme de Fisher et Kedia sont exposées.

4.1.1 Le contexte

L'algorithme de Fisher et Kedia (1990) a été conçu pour résoudre le problème mixte de recouvrement et de partitionnement de tâches. De façon générale, ce type de problème consiste à déterminer un ensemble de groupes de tâches de manière à ce que chacune des tâches considérées soit contenue, dépendamment de son type, dans au moins un des groupes (type recouvrement) ou dans exactement un groupe (type partitionnement). Le regroupement des tâches est régi par certaines règles, invalidant ainsi certains groupes de tâches. Chaque groupe de tâches valide induit un coût, et l'objectif du problème consiste à minimiser le coût total pour le recouvrement / partitionnement des tâches.

De façon mathématique, un tel problème se définit comme suit. Soit $I^1 = \{1, .., n_1\}$ l'ensemble des tâches de type recouvrement, $I^2 = \{1, .., n_2\}$ l'ensemble des tâches de type partitionnement et $J = \{1, .., m\}$ l'ensemble des groupes de tâches valides. Une variable de décision binaire X_j est associée à chaque groupe de tâches valide $j \in J$ de coût c_j. Cette variable prend la valeur 1 si le groupe associé est choisi et 0 sinon. Finalement, notons par I l'union des ensembles I^1 et I^2 (i.e. $I = I^1 \cup I^2$), J_i l'ensemble des groupes valides contenant la tâche $i \in I$ et I_j l'ensemble des tâches contenues dans le groupe $j \in J$. Le problème mixte de recouvrement et partitionnement des tâches se formule alors:

$$\min \sum_{j \in J} c_j X_j \tag{4.1}$$

sous les contraintes

$$\sum_{j \in J_i} X_j \geq 1 \qquad \forall i \in I^1 \tag{4.2}$$

$$\sum_{j \in J_i} X_j = 1 \qquad \forall i \in I^2 \tag{4.3}$$

$$X_j \in \{0,1\} \qquad \forall j \in J. \tag{4.4}$$

L'objectif (4.1) consiste à minimiser le coût total. Les contraintes (4.2) et (4.3) assurent respectivement que les tâches de type recouvrement soient contenues dans au moins un groupe valide, et celles de type partitionnement dans exactement un groupe valide. Finalement, les contraintes (4.4) restreignent les variables de décision à prendre des valeurs binaires.

Il est à noter que pour un groupe j donné, si $I_j \subseteq I^1$, il peut être supposé sans perte de généralité que $c_j > 0$. En effet, si $c_j \leq 0$, alors $X_j = 1$ dans toute solution optimale puisque le problème en est un de minimisation. Ainsi, en posant $X_j = 1$ a priori, la taille du problème peut être réduite. De même, il peut être supposé sans perte de généralité que $I_j \neq \emptyset$, $\forall j \in J$. En effet, dans le cas contraire, il existe un groupe j donné tel que $I_j = \emptyset$. Par conséquent, la valeur de la variable X_j peut être déterminée à l'avance et celle-ci peut être éliminée du problème. Ces deux hypothèses (i.e. $c_j > 0$ $\forall j$ tel que $I_j \subseteq I^1$ et $I_j \neq \emptyset$, $\forall j \in J$) seront retenues pour le restant du chapitre.

Pour les applications qui nous intéressent, I peut représenter un ensemble de parcours d'autobus à effectuer, de clients à servir, ou de vols d'avion nécessitant un

pilote, et J un ensemble d'itinéraires réalisables pour un autobus ou un véhicule, ou d'horaires valides pour un pilote, ou plus généralement, un ensemble de chemins pouvant couvrir les tâches de I. I_j est alors l'ensemble des tâches accomplies au cours du chemin j et c_j est le coût de ce chemin. Le problème est donc de choisir un ensemble de chemins à coût minimum tels que chaque tâche i soit couverte au moins une fois si $i \in I^1$, ou exactement une fois si $i \in I^2$. Dans la suite de ce texte, la terminologie "chemin couvrant des tâches" sera employée au lieu de "groupe de tâches", afin de se rapprocher des applications considérées.

Le problème mixte (4.1)-(4.4) a pour cas particuliers les problèmes de recouvrement de tâches (4.1),(4.2) et (4.4) et de partitionnement de tâches (4.1),(4.3) et (4.4). L'algorithme de Fisher et Kedia semble être le premier à présenter un traitement unifié pour ces deux problèmes. Plusieurs algorithmes ont déjà été développés pour le problème de recouvrement de tâches, utilisant notamment la relaxation lagrangienne et la méthode de sous-gradient (Balas et Ho, 1980). Pour le problème de partitionnement de tâches, le meilleur algorithme semble être celui de Marsten (1974).

Marsten remarque que tous les algorithmes pour résoudre des problèmes de partitionnement de tâches de grande taille commencent par résoudre la relaxation lagrangienne du problème, et c'est ce qui les ralentit. Cela est dû au fait que la relaxation lagrangienne d'un tel problème est très dégénérée, et donc difficile à résoudre. La solution de la relaxation lagrangienne est souvent entière, ou sinon, elle donne une borne inférieure assez serrée pour trouver facilement une solution au problème par un algorithme d'évaluation et de séparation progressive ou par une méthode de coupes. Étant donné la difficulté du calcul, il serait intéressant d'utiliser une alternative à la solution optimale de la relaxation lagrangienne pour trouver de bonnes solutions aux

problèmes de partitionnement ou de recouvrement de tâches.

L'alternative proposée par Fisher et Kedia est d'utiliser plusieurs heuristiques pour trouver une solution optimale ou presque optimale au problème dual (4.1)–(4.4). Ce problème dual s'énonce:

$$\text{Max} \sum_{i \in I} \pi_i \tag{4.5}$$

sous les contraintes

$$\sum_{i \in I_j} \pi_i \leq c_j \qquad \forall j \in J \tag{4.6}$$

$$\pi_i \geq 0 \qquad \forall i \in I^1 \tag{4.7}$$

où π_i est la variable duale correspondant à la contrainte $i \in I$ du problème primal.

La borne supérieure de 1 sur les variables X_j peut être omise dans la relaxation du problème. En effet, soit la condition

$$X_j \geq 0, \ \forall j \in J, \tag{4.8}$$

et soit $X^* = (X_j^* | j \in J)$ une solution optimale de (4.1)–(4.3) et (4.8). Supposons qu'il existe un chemin j donné tel que $X_j^* > 1$. Alors la variable X_j associée à ce chemin ne peut être incluse dans les contraintes de partitionnement (4.3) et se retrouve donc uniquement dans les contraintes de recouvrement (4.2). En remplaçant $X_j = X_j^*$ par $X_j = 1$, une nouvelle solution réalisable de coût inférieur (puisque $c_j > 0$) est obtenue. Comme il s'agit d'un problème de minimisation, cela démontre que X^* ne peut être une solution optimale de (4.1)-(4.3) et (4.8) à moins que $X_j^* \leq 1, \ \forall j \in J$.

L'algorithme de Fisher et Kedia commence par une procédure de type glouton qui trouve une solution initiale pour le problème dual (4.5)-(4.7). Cette solution est

ensuite améliorée par une heuristique 3-opt, qui procède au changement des valeurs de trois variables duales π_i à chaque itération. Éventuellement, si la valeur de la solution duale obtenue n'est pas satisfaisante comparativement à la valeur d'une solution primale trouvée de façon heuristique, une méthode de sous-gradient est utilisée pour trouver une meilleure solution duale. Finalement, une solution entière est obtenue à l'aide d'un processus d'évaluation et de séparation progressive. Tout au long de l'algorithme, des tests logiques simples sont effectués pour vérifier la réalisabilité du problème et en réduire la taille.

Dans les sections qui suivent, la partie duale heuristique de l'algorithme sera décrite, c'est-à-dire la procédure de type glouton (section 4.1.2) et la procédure de type 3-opt (section 4.1.3), et un aperçu de la fin de l'algorithme (section 4.1.4) sera donné. Bien que partie intégrante de l'algorithme de Fisher et Kedia, la méthode de sous-gradient ayant déjà été décrite à la section 2.3, la description de cette méthode ne sera pas reprise. Son intégration à une méthode de génération de colonnes sera toutefois discutée plus loin à la section 4.2. Le lecteur intéressé à une description plus complète de cet algorithme peut consulter l'article de Fisher et Kedia (1990) ou la thèse de Kedia (1985).

Les deux prochaines sections présentent en détail les procédures de type glouton et 3-opt de l'algorithme de Fisher et Kedia. Ces procédures ont été implantées et testées par l'auteur, sans toutefois être retenues pour l'intégration à une méthode de génération de colonnes, comme il est justifié à la section 4.2.1.

4.1.2 La procédure de type glouton

La procédure de type glouton a pour but de trouver une solution initiale pour le problème dual (4.5)-(4.7). La présentation de cette procédure requiert les définitions suivantes:

$$\Delta_i(\pi) = \min_{j \in J_i}(c_j - \sum_{l \in I_j} \pi_l) \tag{4.9}$$

$$I(\pi) = \{i \in I |\ \Delta_i(\pi) > 0\} \tag{4.10}$$

où $\pi = (\pi_i | i \in I)$ est une solution duale réalisable. Pour un vecteur de variables duales π donné, la fonction $\Delta_i(\pi)$ retourne la valeur minimale des coûts réduits des variables associées aux chemins couvrant la tâche i, tandis que $I(\pi)$ donne l'ensemble des tâches pour lesquelles $\Delta_i(\pi)$ retourne une valeur strictement positive, c'est-à-dire l'ensemble des tâches couvertes uniquement par des chemins dont la variable associée a un coût réduit strictement positif.

La procédure de type glouton s'énonce ainsi:

1. - Poser $\pi = 0$.

2. - Pour chaque chemin $j = 1, .., m$, calculer

$$\sigma = \frac{\min\{0, c_j - \sum_{i \in I_j} \pi_i\}}{|I_j \cap I^2|}$$

   et pour toutes les tâches i de $I_j \cap I^2$, poser $\pi_i := \pi_i + \sigma$. (Noter que $\sigma \leq 0$).

3. - Choisir une tâche $i^* \in I(\pi)$ tel que le nombre de chemins couvrant cette tâche $|J_i|$ soit minimisé. S'il y a plusieurs tâches candidates pour i^*, choisir celle qui maximise $\Delta_i(\pi)$. S'il y a encore plusieurs candidates, choisir l'une d'entre elles

arbitrairement.

Poser $\pi_{i^\bullet} := \pi_{i^\bullet} + \Delta_{i^\bullet}(\pi)$.

Mettre à jour $\Delta_i(\pi)$, $\forall i \in I$, et $I(\pi)$.

4. - Arrêter si $I(\pi) = \emptyset$. Sinon, recommencer l'étape 3.

L'étape 1 est une simple initialisation des variables duales π_i à 0. Étant donné que le coût c_j de certains chemins $j \in J$ peut être négatif lorsque $I_j \cap I^2 \neq \emptyset$, cette étape d'initialisation ne garantit pas la validité de cette solution pour le problème dual (4.5)-(4.7). Par conséquent, l'étape 2 permet de vérifier si cette solution duale est réalisable et, si ce n'est pas le cas, d'en construire une qui le soit. Le processus de construction consiste à satisfaire de façon séquentielle chacune des contraintes duales violées en abaissant les valeurs de certaines variables duales $\pi_i, i \in I^2$, ces variables n'étant pas bornées.

Soit j le chemin associé à une des contraintes duales. Cette contrainte duale est satisfaite si

$$\begin{aligned}
& \sum_{i \in (I_j \cap I^2)} (\pi_i + \sigma) + \sum_{i \in (I_j \cap I^1)} \pi_i \leq c_j \\
\Leftrightarrow \quad & \sum_{i \in (I_j \cap I^2)} \sigma \leq c_j - \sum_{i \in I_j} \pi_i \\
\Leftrightarrow \quad & |I_j \cap I^2| \sigma \leq c_j - \sum_{i \in I_j} \pi_i \\
\Leftrightarrow \quad & \sigma \leq \frac{c_j - \sum_{i \in I_j} \pi_i}{|I_j \cap I^2|},
\end{aligned}$$

où $\sigma = \sigma_j(\pi)$ est une quantité, possiblement négative, à additionner à la valeur courante de chacune des variables duales $\pi_i, i \in I_j \cap I^2$. Puisque cette contrainte est déjà satisfaite lorsque $c_j - \sum_{i \in I_j} \pi_i \geq 0$ et que l'ajout d'une quantité positive à la valeur des variables duales $\pi_i, i \in I_j \cap I^2$ pourrait entraîner la violation d'une des contraintes

duales traitées précédemment à cette étape, la quantité $\sigma_j(\pi)$ choisie est donnée par:

$$\sigma_j(\pi) = \frac{\min\{0, c_j - \sum_{i \in I_j} \pi_i\}}{|I_j \cap I^2|}, \tag{4.11}$$

où certaines valeurs $\pi_i, i \in I$ ont pu avoir été modifiées précédemment par des calculs antérieurs de σ_j.

Ainsi, ce choix permet de réduire le moins possible la valeur de la fonction objectif dual tout en assurant la satisfaction de la contrainte duale associée au chemin j.

Il est à noter que l'étape 2 n'est nécessaire que si l'un des coûts c_j est négatif. Ainsi, si $I^2 = \emptyset$, alors $I_j \subseteq I^1$ et $c_j > 0 \; \forall j \in J$: l'étape 2 peut donc être omise. À la fin de l'étape 2 (i.e. après avoir traité séquentiellement chacune des contraintes duales (4.6)), les valeurs attribuées aux variables duales π_i constituent une solution réalisable pour le problème dual (4.5)-(4.7).

L'étape 3 tente d'améliorer la valeur de la fonction objective duale en augmentant le plus possible la valeur de certaines variables duales sans toutefois violer les contraintes. La valeur maximale qui peut être ajoutée à la valeur courante de la variable duale π_i sans violer les contraintes associées aux chemins $j \in J_i$ est donnée par la valeur $\Delta_i(\pi)$ telle que définie par l'équation (4.9). Les autres contraintes duales (i.e. celles associées aux chemins $j \in J \setminus J_i$) ne seront pas violées de toute façon car elles ne contiennent pas la variable π_i.

La tâche i^*, associée à la variable duale π_i qui sera augmentée, est choisie parmi les tâches de l'ensemble $I(\pi)$ tel que défini par (4.10), de façon à minimiser le nombre de contraintes duales $|J_i|$ contenant cette variable. Ce premier critère de sélection permet d'espérer une plus grande augmentation globale de la valeur de la fonction

objectif duale, étant donné que l'augmentation de chaque variable duale a un impact sur un nombre restreint de contraintes duales et qu'ainsi, un plus grand nombre d'augmentations individuelles sera possiblement effectué. En cas d'égalité pour ce critère, la tâche correspondant au $\Delta_i(\pi)$ maximal est choisie, de manière à augmenter le plus possible la variable π_{i^*}. Suite à l'ajustement de la valeur de la variable duale π_{i^*}, les valeurs de $\Delta_i(\pi)$, $\forall i \in I$, et l'ensemble $I(\pi)$ sont mis à jour à l'aide des formules (4.9) et (4.10).

Le critère d'arrêt de la procédure de type glouton est testé à l'étape 4. S'il n'est pas satisfait, l'étape 3 est recommencée jusqu'à ce que $I(\pi) = \emptyset$, c'est-à-dire jusqu'à ce qu'aucune des variables duales π_i ne puisse être augmentée. Une solution heuristique pour le problème dual (4.5)-(4.7) est ainsi obtenue.

4.1.3 La procédure 3-opt

La procédure heuristique 3-opt est une procédure d'amélioration locale de l'objectif qui commence avec la solution duale réalisable trouvée par la procédure de type glouton. Si π_{i_1}, π_{i_2} et π_{i_3} sont trois variables duales, alors π_{i_1} sera diminuée d'une certaine quantité Δ ($\Delta > 0$), tandis que π_{i_2} et π_{i_3} seront augmentées de cette même quantité Δ. Ainsi l'objectif dual (4.5) sera amélioré de Δ. Cette procédure vise à augmenter la valeur de l'objectif en changeant à chaque itération la valeur de trois variables à la fois, tout en préservant la validité de la solution.

Le changement de la valeur des variables duales n'est possible que lorsque certaines conditions sont satisfaites. En partitionnant l'ensemble J des chemins associés aux contraintes du problème dual (4.5)-(4.7) en deux sous-ensembles, soit: le sous-

ensemble des chemins associés aux contraintes actives

$$J^a = \{j \in J | \sum_{i \in I_j} \pi_i = c_i\},$$

et le sous-ensemble des chemins associés aux contraintes inactives

$$J^i = \{j \in J | \sum_{i \in I_j} \pi_i < c_i\},$$

le changement est possible pour une quantité $\Delta > 0$ lorsque:

$$\pi_{i_1} > 0, \text{ si } i_1 \in I^1, \tag{4.12}$$

$$(J_{i_2} \cup J_{i_3}) \cap J^a \subseteq J_{i_1} \text{ et} \tag{4.13}$$

$$J_{i_2} \cap J_{i_3} \cap J^a = \emptyset. \tag{4.14}$$

En effet, la condition (4.12) signifie que π_{i_1} peut être diminuée d'une certaine quantité à déterminer $\Delta > 0$, sans violer la contrainte (4.7); la condition (4.13) indique que, si π_{i_2} ou π_{i_3} se retrouve dans une contrainte active, alors π_{i_1} y est aussi, et ainsi l'augmentation de π_{i_2} ou π_{i_3} pourra être compensée par la diminution de π_{i_1}; et la condition (4.14) implique que toute contrainte active ne peut contenir à la fois π_{i_2} et π_{i_3} car, dans ce cas, la diminution de π_{i_1} ne pourrait compenser pour l'augmentation simultanée de π_{i_2} et π_{i_3}.

Ces conditions sont suffisantes pour que la réassignation désirée des valeurs (c'est-à-dire $\pi_{i_1} := \pi_{i_1} - \Delta, \pi_{i_2} := \pi_{i_2} + \Delta, \pi_{i_3} := \pi_{i_3} + \Delta$) puisse être effectuée, car π_{i_1} peut être diminuée, et π_{i_2} et π_{i_3} augmentées, toutes les contraintes du problème (4.5)-(4.7) restant satisfaites. Il est à noter que certaines contraintes inactives peuvent devenir actives. Il reste donc à décrire la façon de choisir un triplet de variables duales π_{i_1}, π_{i_2} et π_{i_3} satisfaisant (4.5)-(4.7) et à déterminer quelle est la plus grande quantité Δ applicable pour ce triplet.

Pour que la recherche d'un triplet de variables duales π_{i_1}, π_{i_2} et π_{i_3} satisfaisant aux conditions (4.12), (4.13) et (4.14) soit efficace, certaines structures de données sont utilisées. En particulier, les ensembles de chemins J_i sont mis sous la forme de listes chaînées pour permettre de traverser facilement chaque ensemble J_i. D'autre part, des tests spécifiques peuvent être utilisés afin de ne pas avoir à examiner tous les éléments $(i_1, i_2, i_3) \in I \times I \times I$. Certains de ces tests sont étroitement associés à la procédure heuristique primale ou à l'algorithme d'évaluation et de séparation progressive qui sont décrits à la section 4.1.4. Comme expliqué à la section 4.2.1, cette procédure et cet algorithme n'ont pas été retenus dans notre implantation de l'algorithme de Fisher et Kedia. C'est pourquoi les tests qui leur sont associés ne sont pas décrits ici.

D'après Fisher et Kedia, la fouille de $I \times I \times I$ est plus rapide si les conditions (4.12), (4.13) et (4.14) sont vérifiées dans l'ordre, et si le test

$$|J_{i_2} \cap J^a| + |J_{i_3} \cap J^a| \leq |J_{i_1} \cap J^a| \tag{4.15}$$

est inséré entre les tests (4.12) et (4.13). Ce test indique que le nombre de fois que les variables duales π_{i_2} et π_{i_3} se retrouvent dans les contraintes actives ne doit pas dépasser le nombre de fois que la variable π_{i_1} se retrouve dans celles-ci. Ainsi l'augmentation globale de π_{i_2} et π_{i_3} sur l'ensemble des contraintes actives peut être compensée par la diminution globale de π_{i_1} sur cet ensemble de contraintes. Le test (4.15) est toutefois redondant avec les conditions (4.13) et (4.14) puisque

$$\begin{aligned}
& (J_{i_2} \cup J_{i_3}) \cap J^a \subseteq J_{i_1} \text{ [condition (4.13)]} \\
\Rightarrow\ & (J_{i_2} \cup J_{i_3}) \cap J^a \subseteq (J_{i_1} \cap J^a) \text{ [car } (J_{i_2} \cup J_{i_3}) \cap J^a \subseteq J^a \text{]} \\
\Rightarrow\ & (J_{i_2} \cap J^a) \cup (J_{i_3} \cap J^a) \subseteq (J_{i_1} \cap J^a) \\
\Rightarrow\ & |J_{i_2} \cap J^a| + |J_{i_3} \cap J^a| - |(J_{i_2} \cap J^a) \cap (J_{i_3} \cap J^a)| \leq |J_{i_1} \cap J^a|
\end{aligned}$$

[les éléments comptés deux fois sont soustraits]

$$\Rightarrow \quad |J_{i_2} \cap J^a| + |J_{i_3} \cap J^a| - |J_{i_2} \cap J_{i_3} \cap J^a| \leq |J_{i_1} \cap J^a|$$

$$\Rightarrow \quad |J_{i_2} \cap J^a| + |J_{i_3} \cap J^a| \leq |J_{i_1} \cap J^a|$$

[car $J_{i_2} \cap J_{i_3} \cap J^a = \emptyset$ par (4.14)].

Malgré cette redondance, ce test est intéressant car il est plus rapide à effectuer que les tests (4.13) et (4.14), et que dans le cas où il n'est pas satisfait, les tests (4.13) et (4.14) n'ont pas besoin d'être effectués.

Quand un triplet de variables duales π_{i_1}, π_{i_2} et π_{i_3} satisfaisant les conditions (4.12), (4.13) et (4.14) est choisi, les valeurs de ces variables sont modifiées en utilisant la plus grande valeur Δ possible telle que la solution reste réalisable. Pour trouver cette quantité Δ, il est nécessaire d'identifier les contraintes duales inactives associées à un chemin $j \in J^i$ dont la valeur du membre de gauche $(\sum_{i \in I_j} \pi_i)$ augmente à mesure que π_{i_1} diminue et que π_{i_2} et π_{i_3} augmentent.

Dénotons par a_{ij}, pour $i \in I$et $j \in J$, les éléments de la matrice A du problème (4.1)-(4.4):

$$a_{ij} = \begin{cases} 1 & \text{si } i \in I_j \\ 0 & \text{sinon.} \end{cases} \tag{4.16}$$

De plus, dénotons par b la valeur maximale pouvant être soustraite de π_{i_1} sans violer aucune contrainte duale, plus spécifiquement sans violer la contrainte (4.12):

$$b = \begin{cases} \pi_{i_1} & \text{si } i_1 \in I^1 \\ \infty & \text{si } i_1 \in I^2. \end{cases} \tag{4.17}$$

Cette valeur est donc un majorant de Δ. Finalement, soit

$$J^{i^*} = \{j \in J^i |\ a_{i_2 j} + a_{i_3 j} - a_{i_1 j} > 0\} \tag{4.18}$$

l'ensemble des chemins associés aux contraintes duales inactives qui menacent d'être

violées si Δ est trop grand. La contrainte associée à un tel chemin $j \in J^{i^*}$ sera violée si $\sum_{i \in I_j} \pi_i > c_j$.

Lorsque π_{i_2} et π_{i_3} sont augmentées de Δ et que π_{i_1} est diminuée de Δ, le membre de gauche de la contrainte associée au chemin $j \in J^{i^*}$ augmente de $(a_{i_2 j} + a_{i_3 j} - a_{i_1 j})\Delta$. Afin que cette contrainte ne soit pas violée, il faut donc que

$$\sum_{i \in I_j} \pi_i + (a_{i_2 j} + a_{i_3 j} - a_{i_1 j})\Delta \leq c_j$$

$$\Rightarrow \Delta \leq \frac{c_j - \sum_{i \in I_j} \pi_i}{a_{i_2 j} + a_{i_3 j} - a_{i_1 j}}.$$

Puisque cette condition doit être vérifiée pour toutes les contraintes duales inactives associées aux chemins $j \in J^{i^*}$, la quantité Δ est définie par:

$$\Delta = \min \left\{ \min_{j \in J^{i^*}} \left[\frac{c_j - \sum_{i \in I_j} \pi_i}{a_{i_2 j} + a_{i_3 j} - a_{i_1 j}} \right], b \right\}. \tag{4.19}$$

Par la mise à jour des valeurs des variables π_{i_1}, π_{i_2} et π_{i_3}:

$$\pi_{i_1} := \pi_{i_1} - \Delta \tag{4.20}$$

$$\pi_{i_2} := \pi_{i_2} + \Delta \tag{4.21}$$

$$\pi_{i_3} := \pi_{i_3} + \Delta, \tag{4.22}$$

une nouvelle solution duale réalisable de meilleur coût est obtenue. Ce processus itératif d'amélioration local est recommencé jusqu'à ce qu'il n'y ait plus de triplet de variables duales π_{i_1}, π_{i_2} et π_{i_3} satisfaisant les conditions (4.12), (4.13) et (4.14).

La procédure heuristique 3-opt se résume donc ainsi:

1. Trouver un triplet de variables duales π_{i_1}, π_{i_2} et π_{i_3}, $(i_1, i_2, i_3) \in I \times I \times I$, tel que les conditions (4.12), (4.15) (qui est redondante avec les deux suivantes), (4.13) et (4.14) soient vérifiées dans l'ordre;

2. S'il n'existe pas de tel triplet, arrêter l'heuristique 3-opt. Si (i_1, i_2, i_3) existe tel que spécifié, calculer b, J^* et Δ à l'aide des formules (4.17), (4.18) et (4.19).

3. Mettre à jour les valeurs de π_{i_1}, π_{i_2} et π_{i_3} selon les formules (4.20)-(4.22) et retourner à l'étape 1.

4.1.4 Étapes finales de l'algorithme de Fisher et Kedia

Les étapes finales de l'algorithme de Fisher et Kedia consistent en une procédure heuristique primale, un algorithme de sous-gradient et un algorithme d'évaluation et de séparation progressive.

À partir de la solution produite par la procédure 3-opt, une approche heuristique est utilisée pour déterminer une solution au problème primal (4.1)-(4.4). Si la différence entre la valeur de la solution primale heuristique et la borne inférieure trouvée par la procédure heuristique 3-opt est grande (supérieure à 1%), cette borne est améliorée en appliquant une méthode de sous-gradient basée sur celle de Held, Wolfe et Crowder (1974). Cette méthode ne sera pas décrite ici car elle est similaire à celle présentée dans la section 2.3. Il s'agit en fait de la même méthode, restreinte au cas plus simple du recouvrement / partitionnement de tâches. La borne supérieure à la valeur de la solution du problème, nécessaire à une telle méthode de sous-gradient, est donnée par la solution de l'heuristique primale.

L'heuristique primale est basée sur la procédure de type glouton proposée par Chvatàl (1979) pour le problème de recouvrement de tâches, mais où la sélection des variables est basée sur les coûts réduits $\bar{c}_{ij} = c_j - \sum_{i \in I_j} \pi_i$ plutôt que sur les coûts c_j, $\pi = (\pi_i | i \in I)$ étant la solution duale trouvée par les procédures heuristiques de type glouton et 3-opt. L'intégration d'une solution duale dans le calcul d'une solution primale a pour but de mieux calibrer l'impact des contraintes dans la sélection des variables duales.

La procédure heuristique primale commence en posant le vecteur des variables primales $X = (X_j \in \mathbb{R} | j \in J)$ égal à 0. Le principe de cette procédure sera d'augmenter successivement les valeurs de variables primales de 0 à 1, tout en vérifiant que les contraintes primales de recouvrement ne soient pas violées. Le choix de la variable primale X_j à changer se fait comme suit. D'abord, une contrainte primale est choisie de manière à maximiser l'impact du changement, en maximisant la valeur de la variable duale correspondante ainsi que le nombre de variables primales actives de cette contrainte. Parmi les variables actives de la contrainte primale choisie, celle dont le coût réduit est le plus petit et qui est dans le plus grand nombre de contraintes primales est augmentée de 0 à 1. Des variables primales sont ainsi sélectionnées successivement, jusqu'à ce qu'aucun nouveau changement ne soit possible sans violer les contraintes primales de recouvrement. Il est possible que les valeurs des variables primales à la fin de la procédure heuristique primale ne donnent pas une solution réalisable au problème primal (des contraintes de partitionnement pourraient ne pas être satisfaites).

Lorsque le gap entre les solutions primale et duale dépasse 1 %, l'algorithme de sous-gradient est utilisé afin de trouver une meilleure solution heuristique duale.

Dans ce cas, la procédure primale heuristique est ensuite recommencée dans le but de trouver une meilleure solution primale.

Si les valeurs des solutions primale et duale trouvées sont égales, la résolution du problème est terminée, et une solution optimale est donnée par la solution primale. Dans le cas contraire, un algorithme d'évaluation et de séparation progressive est appliqué afin de trouver la solution optimale au problème.

L'algorithme d'évaluation et de séparation progressive utilisé prend une décision de branchement sur une tâche $i \in I$. Pour chaque chemin $j \in J_i$ couvrant cette tâche, un noeud de branchement est créé. Ces noeuds sont ordonnés par ordre croissant du coût réduit et explorés dans cet ordre.

La décision de branchement dépend du type de la tâche $i \in I$ pour laquelle le branchement a été fait. Si i appartient à l'ensemble des tâches devant être couvertes exactement une fois, soit l'ensemble I^1, la décision de branchement pour le noeud défini par le chemin $j \in J_i$ est de poser la valeur de la variable $X_j = 1$ et toutes les autres variables $X_l, l \in J_i$ précédentes, c'est-à-dire dont le coût réduit $\bar{c}_l$ est inférieur au coût réduit $\bar{c}_j$, prennent la valeur 0. Dans le cas où la tâche i appartient à I^2, l'ensemble des tâches devant être couvertes au moins une fois, la décision de branchement pour le noeud défini par le chemin $j \in J_i$ est de poser la valeur de la variable $X_j = 1$, et les autres X_l valent tous 0, pour tous les autres $l \in J_i$.

Les contraintes imposées pour un noeud de branchement définissent un nouveau problème de recouvrement / partitionnement de tâches de taille réduite. Toute la série de procédures heuristiques duales et primale est appliquée à ce nouveau problème. Si l'heuristique primale produit une meilleure solution que toutes les autres solutions

primales trouvées précédemment, la valeur de cette solution est retenue et correspond à une borne supérieure sur la valeur de la solution du problème initial.

L'arbre de branchement est exploré selon une stratégie "meilleur d'abord" ("best-first"). Quand tous les noeuds de branchement ont été résolus ou éliminés selon le cas, l'algorithme de Fisher et Kedia termine et la meilleure solution primale trouvée est optimale.

4.2 Intégration au logiciel GENCOL

Cette section présente l'intégration de l'algorithme de Fisher et Kedia au logiciel GENCOL. Ce logiciel est une implantation informatique de la méthode de génération de colonnes. La section 4.2.1 indique la seule partie de l'algorithme de Fisher et Kedia à avoir été retenue pour implantation dans le logiciel GENCOL, soit l'algorithme de sous-gradient. Les raisons expliquant ce choix y sont aussi énoncées. La section suivante énumère et décrit les nouveaux paramètres qui permettent de choisir l'algorithme de sous-gradient au lieu de l'algorithme du simplexe pour certaines itérations de la méthode de génération de colonnes. Finalement, la section 4.2.3 décrit l'implantation de l'algorithme de sous-gradient dans le logiciel GENCOL ainsi que les nouveaux paramètres propres à cet algorithme.

4.2.1 Ce qui a été retenu de l'algorithme de Fisher et Kedia

Dans une approche de génération de colonnes, la résolution d'une relaxation linéaire consiste en une suite d'itérations composée chacune de la résolution du sous-

problème (génération de colonnes), puis de la résolution d'un problème maître restreint (obtenu par l'addition des nouvelles variables, associées aux colonnes générées, au problème maître restreint de l'itération précédente). Dans la version du logiciel GENCOL utilisée dans le cadre de ce mémoire (la version 4.1b), le problème maître restreint est résolu par le logiciel CPLEX, qui est basé sur l'algorithme du simplexe. Dans la majorité des applications considérées dans le domaine du transport, ce problème maître restreint est très proche d'un problème de recouvrement / partitionnement de tâches. C'est pourquoi une adaptation de l'algorithme de Fisher et Kedia pourrait possiblement remplacer l'algorithme du simplexe pour résoudre de façon heuristique certaines résolutions du problème maître restreint de la méthode de génération de colonnes, afin de réduire le temps total de résolution. Cette section indiquera comment la substitution de l'algorithme du simplexe par l'algorithme de Fisher et Kedia est faite.

La première résolution du problème maître restreint sera toujours faite par l'algorithme du simplexe. Pour les itérations suivantes de la méthode de génération de colonnes, le problème maître restreint sera résolu au choix par l'algorithme du simplexe ou par l'algorithme de Fisher et Kedia modifié. De cet algorithme n'est gardé que l'algorithme de sous-gradient. En effet, la procédure de type glouton n'est plus nécessaire car une solution duale de départ pour le problème maître restreint est disponible. Cette solution duale est donnée par les variables duales trouvées à l'itération précédente de la méthode de génération de colonnes, dont le problème maître restreint a été résolu soit par l'algorithme du simplexe, soit par l'algorithme de sous-gradient. De plus, des tests préliminaires sur l'algorithme de Fisher et Kedia ont démontré que le temps requis par la procédure 3-opt était beaucoup trop long pour la rapidité demandée à une heuristique, ceci malgré les efforts qui ont été mis sur la qualité

de l'implantation. Ce résultat contredit les conclusions de Fisher et Kedia qui attribuaient la rapidité de leur algorithme à cette procédure 3-opt!

L'heuristique primale proposée par Fisher et Kedia, déterminant une solution primale, n'est plus nécessaire non plus. La valeur de cette solution fournit une borne supérieure utilisée par l'algorithme de sous-gradient. Or, une telle borne est déjà disponible: elle est donnée par la dernière solution primale trouvée par l'algorithme du simplexe dans les itérations précédentes de la méthode de génération de colonnes. Si les coefficients de toutes les variables dans la fonction objectif ne changent pas durant la résolution, comme c'est le cas en général, cette solution est alors la meilleure trouvée jusque là. Par contre, lorsque l'algorithme utilisé fait varier les coefficients de certaines variables dans la fonction objectif (tel l'algorithme décrit au chapitre 5), la dernière solution primale trouvée par l'algorithme du simplexe ne procure plus nécessairement une borne valide. Cette borne potentiellement non valide sera tout de même employée lorsque l'algorithme de sous-gradient sera combiné à une stratégie de perturbation au chapitre 5, puisque les perturbations considérées seront négligeables.

Enfin, l'algorithme d'évaluation et de séparation progressive proposé par Fisher et Kedia est aussi écarté, car son but est de trouver une solution optimale entière à partir de la solution trouvée par l'heuristique duale. Puisque la présente étude se limite à la résolution de la relaxation linéaire du problème, cet algorithme n'est pas nécessaire dans notre cas.

De l'algorithme de Fisher et Kedia, il ne reste donc que l'algorithme de sous-gradient, généralisé comme il a été vu à la section 2.3. Cet algorithme s'intègre aisément à la méthode de génération de colonnes, en particulier pour le logiciel

GENCOL. À une itération donnée de la méthode de génération de colonnes, le problème maître restreint est résolu en utilisant soit l'algorithme du simplexe (primal ou dual), soit l'algorithme de sous-gradient. Le choix de l'algorithme à utiliser est régi par différents paramètres, décrits dans la section suivante. D'autres paramètres, dont il est question à la section 4.2.3, permettent d'ajuster la méthode de sous-gradient même.

Notons que les contraintes de disponibilité des véhicules (contraintes 3.10) du problème maître restreint sont traitées à part dans l'implantation proposée. L'algorithme de sous-gradient étant valide pour un problème maître général, et non seulement un problème de recouvrement / partitionnement de tâches, il peut s'appliquer aisément au cas considéré. Mais ayant constaté que les variables duales associées à ces contraintes particulières s'ajustaient beaucoup plus rapidement que les autres et demeuraient stables par la suite, nous avons décidé de ne pas faire agir l'algorithme de sous-gradient sur ces variables (seul l'algorithme du simplexe peut réajuster leurs valeurs).

4.2.2 Les paramètres de choix de l'algorithme de sous-gradient

Les paramètres de choix de l'algorithme de sous-gradient permettent de déterminer, à chaque itération de la méthode de génération de colonnes, si le problème maître restreint sera résolu avec l'algorithme du simplexe (approche standard) ou avec l'algorithme de sous-gradient (nouvelle approche). Ce choix dépendra entre autres du type d'itération (l'itération est dite en phase II si une solution primale réalisable a déjà été trouvée, sinon elle est dite en phase I), de la méthode choisie pour résoudre

le problème maître des itérations précédentes, et du nombre de colonnes générées à l'itération courante. La présente section donne la liste des paramètres régissant ce choix.

Le premier paramètre, **SgUse**, est booléen et indique si l'algorithme de sous-gradient peut être employé pour le problème donné. Lorsque **SgUse** = 0, tous les problèmes maîtres restreints sont résolus par l'algorithme du simplexe, comme dans la version standard de l'optimiseur GENCOL. Dans le cas contraire, l'algorithme de sous-gradient peut être appelé à remplacer l'algorithme du simplexe selon le processus de résolution et les valeurs des autres paramètres. La valeur par défaut du paramètre **SgUse** est 1.

Le paramètre booléen **SgPhaseI** spécifie si l'algorithme de sous-gradient peut être utilisé (**SgPhaseI** = 1) ou non (**SgPhaseI** = 0) pour les itérations de la phase I de la résolution du problème, c'est-à-dire avant d'avoir trouvé une solution primale réalisable. La valeur par défaut de ce paramètre a été fixée à 0. Quelques tests ont en effet montré qu'il était avantageux d'éviter l'utilisation de l'algorithme de sous-gradient pendant la phase I. Ce résultat pourrait toutefois être dû au fait que la première itération est toujours accomplie avec l'algorithme du simplexe. En général, dans les premières itérations de la méthode de génération de colonnes à l'intérieur du logiciel GENCOL, les solutions obtenues comportent des variables artificielles de très grand coût, qui disparaissent assez rapidement en cours de résolution. Ces coûts très grands induisent des valeurs de départ très grandes pour les variables duales. Dans un tel contexte, l'algorithme de sous-gradient s'avère peu performant étant donné que l'information duale initiale est peu pertinente. Pour pallier à cette difficulté, il faudrait fournir au logiciel GENCOL des valeurs de départ plausibles pour les

variables duales. Comme la recherche de telles valeurs initiales constitue lui-même un problème complexe, cette avenue n'a pas été explorée dans ce projet.

Le paramètre entier **SgIterConsMax** donne un maximum sur le nombre d'itérations consécutives de la méthode de génération de colonnes pour lesquelles l'algorithme de sous-gradient est utilisé. En d'autres mots, lors de la résolution, si l'algorithme de sous-gradient a été utilisé pour résoudre le problème maître restreint des **SgIterConsMax** dernières itérations, le problème maître restreint de la présente itération sera résolu avec l'algorithme du simplexe. Ceci a pour but d'assurer un certain contrôle sur les valeurs prises par les variables duales à l'aide de l'algorithme du simplexe. En effet, puisque l'algorithme de sous-gradient est heuristique, il n'y a aucune raison de croire qu'en utilisant cet algorithme à chaque résolution d'un problème maître restreint, la solution du problème maître convergera vers la solution optimale de la relaxation linéaire. De plus, la résolution primale du problème maître restreint permet de remettre à jour la borne supérieure utilisée dans l'algorithme de sous-gradient. Par défaut, le paramètre **SgIterConsMax** vaut 12.

Quand de nouvelles colonnes sont générées, des variables primales associées à ces colonnes sont ajoutées au problème maître restreint. Or, afin de réduire le temps requis pour calculer les coûts réduits des variables hors base (resp. un sous-gradient) lorsque l'algorithme du simplexe (resp. l'algorithme de sous-gradient) est employé, une limite sur le nombre de variables considérées par le problème maître restreint peut être imposée dans le logiciel GENCOL. Lorsque cette limite est atteinte (c'est-à-dire lorsque trop de colonnes ont été générées), certaines variables doivent être éliminées: cette opération est surnommée "faire le ménage". Il est à noter que cette stratégie d'accélération ne compromet pas l'optimalité de la méthode de résolution,

étant donné que les colonnes associées aux variables rejetées peuvent être générées à nouveau dans les itérations subséquentes. Le paramètre booléen **SgMenageUse** sert à imposer l'utilisation de l'algorithme du simplexe quand le ménage vient d'être fait (**SgMenageUse** = 1), ou à laisser les autres paramètres choisir entre cet algorithme et l'algorithme de sous-gradient dans un tel cas (**SgMenageUse** = 0). Dans ce dernier cas, beaucoup de colonnes peuvent être éliminées avant qu'une nouvelle résolution du problème maître restreint par l'algorithme du simplexe survienne. Dans notre implantation, éliminer un trop grand nombre de variables primales paralysait l'algorithme du simplexe. Pour remédier à cette situation, il devient nécessaire que l'algorithme du simplexe recalcule une nouvelle base à partir des variables disponibles, lorsque le paramètre **SgMenageUse** = 0. C'est pourquoi la valeur par défaut de **SgMenageUse** est 1. Sinon, si l'algorithme de sous-gradient est utilisé trop souvent, le temps gagné à éviter l'algorithme du simplexe est repris dans des calculs plus lourds lorsque l'algorithme du simplexe est appelé.

Le paramètre entier **SgMinColGen** donne une limite inférieure sur le nombre de colonnes générées à l'itération courante à partir de laquelle l'utilisation de l'algorithme de sous-gradient est permise pour résoudre le problème maître restreint. En effet, il a été observé en général que l'algorithme du simplexe est plus rapide que celui de sous-gradient lorsque peu de colonnes sont générées. Dans ce cas il n'y a donc aucun intérêt à choisir l'algorithme de sous-gradient. Le paramètre **SgMinColGen** est aussi utile en fin de résolution de la relaxation linéaire, lorsque peu de colonnes sont générées à chaque itération: la méthode de sous-gradient n'étant pas optimale, il est nécessaire de terminer la résolution à l'aide de l'algorithme du simplexe. En particulier, même si les variables duales fournies par l'algorithme de sous-gradient ne permettent pas de générer des variables à coût réduit négatif, le processus de résolution

ne doit pas s'arrêter à cette itération et le dernier problème maître restreint doit être résolu à nouveau en utilisant une méthode optimale telle l'algorithme du simplexe. **SgMinColGen** doit donc être au moins égal à 1. Le paramètre **SgMinColGen** vaut 6 par défaut.

Enfin, le paramètre entier **SgNbIterGenMax** indique un nombre maximum d'itérations de la méthode de génération de colonnes pour lesquelles le problème maître restreint est résolu par l'algorithme de sous-gradient: à partir de l'itération **SgNbIterGenMax**+1 de la méthode de génération de colonnes, le problème maître restreint est toujours résolu avec l'algorithme du simplexe. Par défaut, **SgNbIterGenMax** = 0: dans ce cas, ce paramètre est désactivé.

Pour une itération donnée de la méthode de génération de colonnes dans le logiciel GENCOL, le problème maître restreint n'est résolu avec l'algorithme de sous-gradient que si tous les paramètres venant d'être décrits le permettent. Sinon, l'optimiseur standard pour le problème maître restreint sera utilisé, soit l'optimiseur CPLEX dans la version courante du logiciel GENCOL.

4.2.3 L'algorithme de sous-gradient dans le logiciel GENCOL

Cette section décrit comment l'algorithme de sous-gradient, présenté dans la section 2.3, est implanté dans le logiciel GENCOL. Elle introduit aussi les divers paramètres spécifiques à cet algorithme.

La description de l'implantation de l'algorithme de sous-gradient dans le logiciel GENCOL s'inscrit dans le contexte suivant. Soit g le numéro d'une itération de la

méthode de génération de colonnes du logiciel GENCOL pour laquelle de nouvelles colonnes de coût réduit négatif viennent d'être générées, et de nouvelles variables associées à ces colonnes ajoutées au problème maître restreint. Étant donné les valeurs attribuées aux paramètres de choix, ce nouveau problème maître restreint doit être résolu en utilisant l'algorithme de sous-gradient, afin d'ajuster les valeurs des variables duales trouvées à l'itération précédente $(g-1)$ de la méthode de génération de colonnes.

Pour résoudre le problème maître restreint de l'itération g, S_g itérations de l'algorithme de sous-gradient seront effectuées. Le vecteur des variables duales, $\pi_g^s = (\pi_{g,1}^s, \pi_{g,2}^s, ..., \pi_{g,n}^s)$, est initialisé à π_g^0 au début de l'algorithme de sous-gradient, et d'une itération s à l'itération suivante $s+1$ de cet algorithme, ce vecteur passe de π_g^s à π_g^{s+1}. Lorsqu'il n'y aura aucun risque de confusion, le vecteur des variables duales sera simplement noté π^s. D'une itération g à l'itération suivante $g+1$ de la méthode de génération de colonnes, le vecteur des variables duales π passe de π_g à π_{g+1} avec $\pi_g = \pi_g^0$ et $\pi_g^{S_g} = \pi_{g+1}^0$.

Le vecteur de départ $\pi^0 = \pi_g^0$ pourrait être trouvé par une procédure de type glouton similaire à celle employée par Fisher et Kedia (1990), ou à celle de Toth et al. (1996). Mais la procédure de type glouton de Fisher et Kedia n'est plus nécessaire car une solution duale de départ est donnée par les variables duales trouvées à l'itération précédente de la méthode de génération de colonnes. Pour la même raison, la procédure de Toth et al. n'a pas été implantée ni testée. Le vecteur de départ provient donc de la résolution du problème maître restreint de l'itération précédente de la méthode de génération de colonnes (soit par l'algorithme du simplexe, soit par l'algorithme de sous-gradient). Ce vecteur de départ est non réalisable pour le problème

dual associé au nouveau problème maître restreint car les nouvelles contraintes, celles associées aux variables de coût réduit négatif ajoutées au problème maître restreint, sont nécessairement violées par cette solution duale. Or, d'après Held, Wolfe et Crowder (1974), les solutions obtenues par la méthode de sous-gradient varient peu selon les valeurs du vecteur de départ de l'algorithme, tant que celles-ci ne sont pas trop mauvaises.

À chaque itération s de l'algorithme de sous-gradient, la direction d^s (vers où π^s doit se diriger) et le pas θ^s (la longueur du déplacement dans la direction d^s) doivent être calculés. La borne supérieure $\overline{Z}_{LR}$ nécessaire au calcul du pas est donnée par la valeur de la meilleure solution primale trouvée par l'algorithme du simplexe dans les itérations précédentes de la méthode de génération de colonnes. Comme décrit dans la section 2.3, θ^s dépend d'un paramètre ρ^s qui assure la convergence de l'algorithme. La valeur initiale ρ^0 est 1. La valeur de ce paramètre peut varier en cours de résolution et est déterminée à l'aide de deux paramètres du logiciel GENCOL: le nombre entier **SgNbItDivRho** et le nombre réel **SgFactDivRho**. À toutes les **SgNbItDivRho** itérations de l'algorithme de sous-gradient, ρ^s est divisé par **SgFactDivRho**. Par défaut, ces paramètres valent 20 et 2, respectivement.

Dans le logiciel GENCOL, la résolution du problème maître restreint de l'itération g par l'algorithme de sous-gradient se termine lorsqu'un des critères d'arrêt suivants est satisfait:

1. la valeur de la fonction duale, $L_D(\pi_g^s)$, a atteint la valeur de la borne supérieure $\overline{Z}_{LR}$, au gap admissible près donné (en %) par le paramètre réel **SgMinGap**, qui vaut par défaut 0,5;

2. le nombre d'itérations de l'algorithme de sous-gradient S_g a atteint une borne supérieure donnée par le paramètre entier **SgNbIterMax** qui, par défaut, vaut 400;

3. le paramètre ρ^s est devenu tellement petit que le pas θ^s est pratiquement nul ($\rho^s \leq \epsilon$ pour le ϵ donné par le paramètre du logiciel GENCOL **MpEpsOpt** valant par défaut 10^{-6}).

4.3 Résultats

Une batterie de tests sur les paramètres de l'algorithme de sous-gradient ont permis de vérifier l'efficacité de l'intégration de cet algorithme dans une méthode de génération de colonnes. Ces tests ont porté sur la résolution de la relaxation linéaire de problèmes de type m-TSPTW tel que décrits au chapitre 3, et comportant jusqu'à 600 tâches. Les tests ont été effectués sur un ordinateur de type hp9000/715, à l'aide de la version 4.1b de l'optimiseur GENCOL.

Le tableau 4.1 présente des résultats numériques sur un ensemble de 25 problèmes. La première colonne identifie le problème. Le premier nombre de l'identificateur indique le nombre de tâches considérées dans le problème. Le deuxième nombre de cet identificateur est un numéro donné au problème. Les deuxième et troisième colonnes indiquent les temps totaux de résolution (en secondes) obtenus en utilisant la version standard de GENCOL et la version modifiée de GENCOL intégrant l'algorithme de sous-gradient, respectivement. Cette version modifiée utilise les valeurs par défaut des nouveaux paramètres. Finalement, la dernière colonne donne l'amélioration relative (en pourcentage) obtenue par la version modifiée, soit $\frac{T_S - T_M}{T_S}$, où T_S et T_M représentent

les temps de résolution des versions standard et modifiée, respectivement.

Ces résultats montrent que pour des problèmes de petite taille (100 ou 150 tâches), l'algorithme de sous-gradient n'est pas avantageux. Sur les 8 problèmes résolus, l'amélioration observée varie de -24,6 % à 10,2 %. En moyenne, le temps de résolution est 9,5 % plus rapide avec la version standard. Ces résultats sont imputables au fait que pour des problèmes de petite taille, l'algorithme du simplexe est très performant.

Par contre, dès que la taille des problèmes devient plus importante (500 ou 600 tâches), l'algorithme de sous-gradient devient utile. C'est qu'en comparaison, l'algorithme du simplexe perd beaucoup de temps dans de trop fréquentes réoptimisations. Du temps est perdu à résoudre des problèmes maîtres restreints jusqu'à l'optimalité, alors que la solution de la relaxation linéaire est si éloignée qu'une solution heuristique serait amplement suffisante. Sur les 17 problèmes traités, le temps de résolution avec la version modifiée est de 12,0 % à 41,7 % plus rapide qu'avec la version standard. En moyenne, il est de 28,0 % plus rapide sur les problèmes étudiés.

Des tests ont été faits plus spécifiquement sur l'ajustement des paramètres de l'algorithme de sous-gradient. Cette étude de la sensibilité des nouveaux paramètres se divise en trois tableaux.

- Le tableau 4.2 porte sur des variations de paramètres qui augmentent la fréquence d'utilisation de l'algorithme de sous-gradient pour résoudre le problème maître restreint.
- Le tableau 4.3 porte sur des variations de paramètres qui diminuent la fréquence d'utilisation de cet algorithme.

Tableau 4.1: Temps de résolution avec ou sans l'algorithme de sous-gradient

problème	**temps standard (s)**	**temps avec sous-gradient (s)**	**amélioration (%)**
150-4	25,2	31,4	-24,6
100-2	6,5	7,9	-21,5
100-3	7,1	8,6	-21,1
100-4	7,5	9,0	-20,0
100-1	8,9	9,3	-4,5
150-2	30,6	30,8	-0,7
150-3	29,9	28,0	6,4
150-1	42,3	38,0	10,2
600-2	1265,0	1112,8	12,0
600-3	1416,2	1109,6	21,6
600-1	1834,0	1423,6	22,4
600-7	1661,1	1274,9	23,2
600-12	2001,0	1516,1	24,2
500-5	936,4	699,8	25,3
600-5	1772,9	1283,4	27,6
600-10	1730,3	1221,1	29,4
600-6	1673,6	1174,6	29,8
600-4	1815,9	1237,4	31,9
600-8	1748,2	1177,7	32,6
600-11	1827,2	1220,6	33,2
500-4	895,3	595,1	33,5
500-3	1093,1	696,0	36,3
500-2	1190,6	730,1	38,7
600-9	2306,5	1385,2	39,9
500-1	1344,0	783,3	41,7

- Enfin, le tableau 4.4 porte sur des variations de paramètres qui contrôlent directement l'algorithme de sous-gradient.

Les tableaux 4.2 à 4.4 sont tous structurés de la même façon. Chaque ligne de résultats donne les temps de résolution minimum et maximum (en secondes) obtenus pour 10 problèmes de 600 tâches, ainsi que la moyenne et l'écart-type de ces temps de résolution.

Chacun de ces tableaux comprend d'abord les résultats de la résolution sans l'algorithme de sous-gradient. Il comprend ensuite les résultats pour une résolution de ces problèmes utilisant l'algorithme de sous-gradient, avec les valeurs de défaut des paramètres associés à cet algorithme. Ce sont de bonnes valeurs pour les paramètres, qui ont été choisies après de nombreux essais. Les résultats associés à ces valeurs de défaut constituent donc une référence valable.

Finalement, chaque tableau comprend les résultats obtenus en modifiant la valeur d'un paramètre, tandis les autres gardent leur valeur par défaut.

Dans les tests du tableau 4.2, la valeur de certains paramètres a été modifiée de façon à ce que l'algorithme de sous-gradient soit utilisé plus souvent pour résoudre le problème maître restreint. Le nombre maximum d'itérations consécutives de l'algorithme de génération de colonnes utilisant l'algorithme de sous-gradient, est passé de 12 à 24, en changeant la valeur du paramètre **SgIterConsMax**. Le nombre minimum de colonnes demandé pour permettre l'application de l'algorithme de sous-gradient, **SgMinColGen**, fut diminué de 12 à 3. L'algorithme de sous-gradient a été appliqué durant la phase I (avant qu'une solution fractionnaire ne soit disponible) grâce au

paramètre **SgPhaseI**. Enfin, l'algorithme de sous-gradient fut utilisé plus souvent en permettant son utilisation même dans les cas où le ménage vient d'être fait, en donnant la valeur 0 au paramètre **SgMenageUse**. Dans ce dernier cas, lorsqu'une résolution du problème maître restreint avec l'algorithme du simplexe survient après une suite de résolutions utilisant l'algorithme de sous-gradient, lors desquelles au moins un ménage des variables duales a été fait, l'algorithme du simplexe recompile une nouvelle base.

Tableau 4.2: Temps de résolution avec une utilisation plus fréquente de l'algorithme de sous-gradient

paramètres	min	max	moyenne	écart-type
Sans sous-gradient	**1265,0**	**2306,5**	**1722,4**	**259,8**
Valeurs par défaut	**1109,6**	**1423,6**	**1240,0**	**99,8**
SgIterConsMax = 24	1088,5	1464,7	1242,6	117,7
SgMinColGen = 3	1113,9	1467,0	1252,7	116,0
SgPhaseI = 1	1157,3	1569,6	1400,1	126,4
SgMenageUse = 0	1074,0	8666,4	2110,6	2202,2

L'augmentation de la valeur du paramètre **SgIterConsMax** ou la diminution de la valeur du paramètre **SgMinColGen** ne font que ralentir très légèrement la résolution. L'application du paramètre **SgPhaseI** ralentit la résolution de manière plus marquée. Enfin, en n'appliquant pas le paramètre **SgMenageUse**, le temps gagné par l'application plus fréquente de l'algorithme de sous-gradient est perdu au prochain appel de l'algorithme du simplexe, lors duquel une nouvelle base doit généralement être calculée, ce qui consomme beaucoup de temps. De manière générale, malgré l'efficacité de l'algorithme de sous-gradient, il semble bon de ne pas l'utiliser plus fréquemment que ce que permettent les valeurs par défaut des paramètres correspondants.

Les valeurs de paramètres testées pour le tableau 4.3 diminuent la fréquence d'uti-

lisation de l'algorithme de sous-gradient par rapport à l'algorithme du simplexe, pour la résolution du problème maître restreint de chaque itération de génération de colonnes. Cette fois-ci, le nombre minimum exigé de colonnes générées pour permettre l'utilisation de l'algorithme de sous-gradient (paramètre **SgMinColGen**) passe de 6 à 20. Dans le test suivant, le nombre maximum d'itérations consécutives de génération de colonnes utilisant l'algorithme de sous-gradient, donné par le paramètre **SgIterConsMax**, est diminué de 12 à 3.

Tableau 4.3: Temps de résolution avec une utilisation plus rare de l'algorithme de sous-gradient

paramètres	min	max	moyenne	écart-type
Sans sous-gradient	**1265,0**	**2306,5**	**1722,4**	**259,8**
Valeurs par défaut	**1109,6**	**1423,6**	**1240,0**	**99,8**
SgMinColGen = 20	1042,3	1465,5	1236,9	129,5
SgIterConsMax = 3	1107,6	1622,0	1361,5	150,1

L'augmentation de **SgMinColGen** accélère un tout petit peu la résolution par rapport aux valeurs par défaut. De son côté, le test effectué en diminuant le paramètre **SgIterConsMax** ralentit la résolution par rapport aux valeurs standard. Ceci confirme que les valeurs par défaut des paramètres déterminant la fréquence d'utilisation de l'algorithme de sous-gradient par rapport à l'algorithme du simplexe sont raisonnables.

Dans le tableau 4.4, les paramètres qui touchent la manière dont est appliqué l'algorithme de sous-gradient lors de la résolution d'un problème maître restreint sont testés. Il s'agit, d'une part, des paramètres contrôlant la convergence de l'algorithme de sous-gradient pendant son application. Rappelons que cette convergence est accélérée par le biais du facteur ρ, celui-ci étant divisé par la valeur **SgFactDivRho** à chaque **SgNbItDivRho** itérations de l'algorithme de sous-gradient. Cette conver-

gence sera accélérée en diminuant la valeur du paramètre **SgNbItDivRho** de 20 à 8.

D'autre part, il s'agit de paramètres indiquant des critères d'arrêt de l'algorithme de sous-gradient. Le nombre maximum d'itérations de l'algorithme de sous-gradient pendant la résolution d'un problème maître restreint est réduit de 400 à 200 par le biais du paramètre **SgNbIterMax**. Le gap en-dessous duquel l'algorithme de sous-gradient est arrêté, gap (en %) entre la valeur de l'objectif dual trouvée par l'algorithme de sous-gradient, et sa borne supérieure connue, est donné par la valeur du paramètre **SgMinGap**. La valeur par défaut de ce paramètre est 0,5, et les valeurs testées 1 et 0. Ces valeurs correspondent dans le premier cas à un arrêt plus rapide de l'algorithme, dans le deuxième cas à un algorithme de sous-gradient poussé à l'optimalité (si les autres critères d'arrêt ne sont pas atteints).

Tableau 4.4: Temps de résolution selon les paramètres internes à l'algorithme de sous-gradient

paramètres	min	max	moyenne	écart-type
Sans sous-gradient	**1265,0**	**2306,5**	**1722,4**	**259,8**
Valeurs par défaut	**1109,6**	**1423,6**	**1240,0**	**99,8**
SgNbIterMax = 200	1049,5	1375,7	1222,9	105,2
SgNbItDivRho = 8	1064,2	1386,4	1225,2	98,6
SgMinGap = 1	1052,4	1576,2	1297,1	148,7
SgMinGap = 0	1901,2	>44000	≈14045,5	≈13749,4

La réduction du paramètre **SgNbIterMax** accélère légèrement la résolution. De même, la diminution du paramètre **SgNbItDivRho** accélère aussi la résolution par rapport aux valeurs par défaut. Lorsque le paramètre **SgMinGap** vaut 1, l'algorithme de sous-gradient s'arrête plus rapidement, avec une moins bonne solution duale. Cette solution ne doit pas être assez bonne pour permettre de générer de bonnes colonnes, car le temps de résolution total est alors diminué. Au contraire, quand le paramètre

SgMinGap vaut 0, l'algorithme de sous-gradient ne s'arrête que s'il trouve une solution duale de même valeur que la borne supérieure donnée, ou si un autre critère d'arrêt est atteint. L'algorithme perd alors beaucoup de temps à préciser une solution qui n'a qu'une valeur heuristique, et les temps de résolution deviennent démesurés (certains tests ont même du être arrêtés avant leur résolution complète). Cette valeur est à déconseiller pour ce paramètre. En résumé, il peut être avantageux de faire converger l'algorithme de sous-gradient plus rapidement par l'entremise du facteur ρ et des paramètres **SgNbIterMax** et **SgNbItDivRho**. Au contraire, le paramètre **SgMinGap** est très sensible, et sa valeur par défaut semble adéquate.

Il est intéressant de noter que dans tous les tests sur les paramètres sauf pour le cas très mauvais où le paramètre **SgMinGap** vaut 0, l'écart-type des temps de résolution est plus petit lorsque l'algorithme de sous-gradient est utilisé qu'avec la résolution par défaut. Les temps de résolution sont donc plus homogènes lorsque l'algorithme de sous-gradient est utilisé.

En résumé pour l'ajustement des paramètres, il est avantageux d'utiliser l'algorithme de sous-gradient, modéré d'une intervention régulière de l'algorithme du simplexe. Il est efficace que cette intervention soit faite à chaque fois qu'un ménage est effectué au niveau des variables primales du problème maître restreint. La combinaison idéale d'algorithmes est d'utiliser souvent l'algorithme de sous-gradient au début de la résolution, lorsque des valeurs heuristiques sont suffisantes et beaucoup plus rapides à obtenir. À la fin de la résolution, lorsque le gap entre la valeur de la solution duale donnée par l'algorithme de sous-gradient et la valeur de la meilleure solution primale connue devient petit, il devient avantageux de n'utiliser que l'algorithme du simplexe.

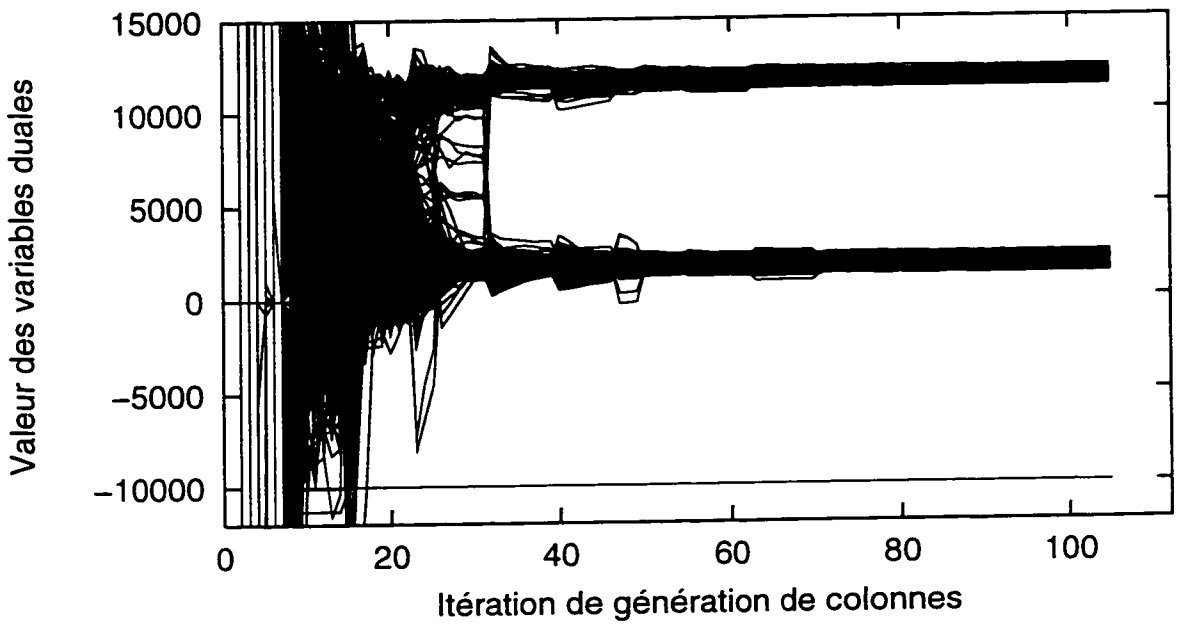


Figure 4.1: Comportement des variables duales avec l'algorithme de sous-gradient

La figure 4.1 représente l'évolution des valeurs des variables duales pendant la résolution d'un problème à l'aide de l'algorithme de sous-gradient. Cette résolution utilise les valeurs par défaut des paramètres de sous-gradient. Dans cette figure, le numéro de l'itération est en abscisse, et les valeurs des variables duales en ordonnée. Notons que ce type de graphe n'est pas une bonne indication de l'évolution des variables duales au cours du temps, car une itération résolue à l'aide de l'algorithme de sous-gradient est généralement beaucoup plus rapide qu'une itération résolue à l'aide de l'algorithme du simplexe. Il est intéressant de comparer l'évolution de la résolution avec la figure 1.1 du chapitre 1, représentant la résolution du même problème sur la même échelle (valeurs des variables duales en fonction de l'itération de génération de colonnes), avec l'algorithme standard de résolution.

Comment l'application de l'algorithme de sous-gradient influence-t-il l'évolution de la convergence des variables duales? Cette évolution se fait en plusieurs parties.

D'abord a lieu la phase I, pendant laquelle des variables artificielles sont encore actives dans le problème maître restreint. Rappelons que les variables artificielles ont un coût très grand, ce qui rend les variables duales elles-mêmes très grandes (de l'ordre de 10^{20}). Pendant cette phase, l'algorithme de sous-gradient n'est pas appliqué, soit lors des itérations 1 à 7 pour l'exemple considéré.

En deuxième lieu, des itérations 8 à 32, a lieu une phase très chaotique lors de laquelle les valeurs des variables duales varient énormément d'une itération à l'autre. Puis, des itérations 33 à 105, les variables duales sont plus proches de leurs valeurs finales et ne font que s'ajuster légèrement. L'algorithme de sous-gradient est appliqué entre les itérations 8 et 70 inclusivement, en moyenne 5 itérations sur 6 (notamment à chaque fois qu'un "ménage" est fait parmi les variables gardées pour le problème maître restreint). Par la suite, seul l'algorithme du simplexe est utilisé pour résoudre le problème maître restreint, car le gap entre la solution duale et la borne supérieure donnée par la dernière solution primale trouvée est toujours plus petit que le gap permis par le paramètre **SgMinGap**.

Les itérations dans lesquelles l'algorithme du simplexe est utilisé se reconnaissent bien par rapport aux itérations dans lesquelles l'algorithme de sous-gradient est utilisé, par les soubresauts qu'elles impriment au graphe.

Il semble que l'algorithme de sous-gradient ait le plus d'influence pendant la période dite "chaotique", qui va des itérations 8 à 32. C'est pendant cette période que la résolution avec l'algorithme de sous-gradient prend le plus d'avance sur la résolution standard, en termes de valeur de l'objectif par rapport au temps total de résolution (voir la figure 4.2). Pendant cette période, il importe peu de résoudre le

problème maître restreint jusqu'à l'optimalité, car la solution partielle est encore très loin de la solution optimale. Par la suite, moins de colonnes sont générées et l'algorithme du simplexe n'a plus besoin de l'aide de l'algorithme de sous-gradient car les réoptimisations sont très rapides.

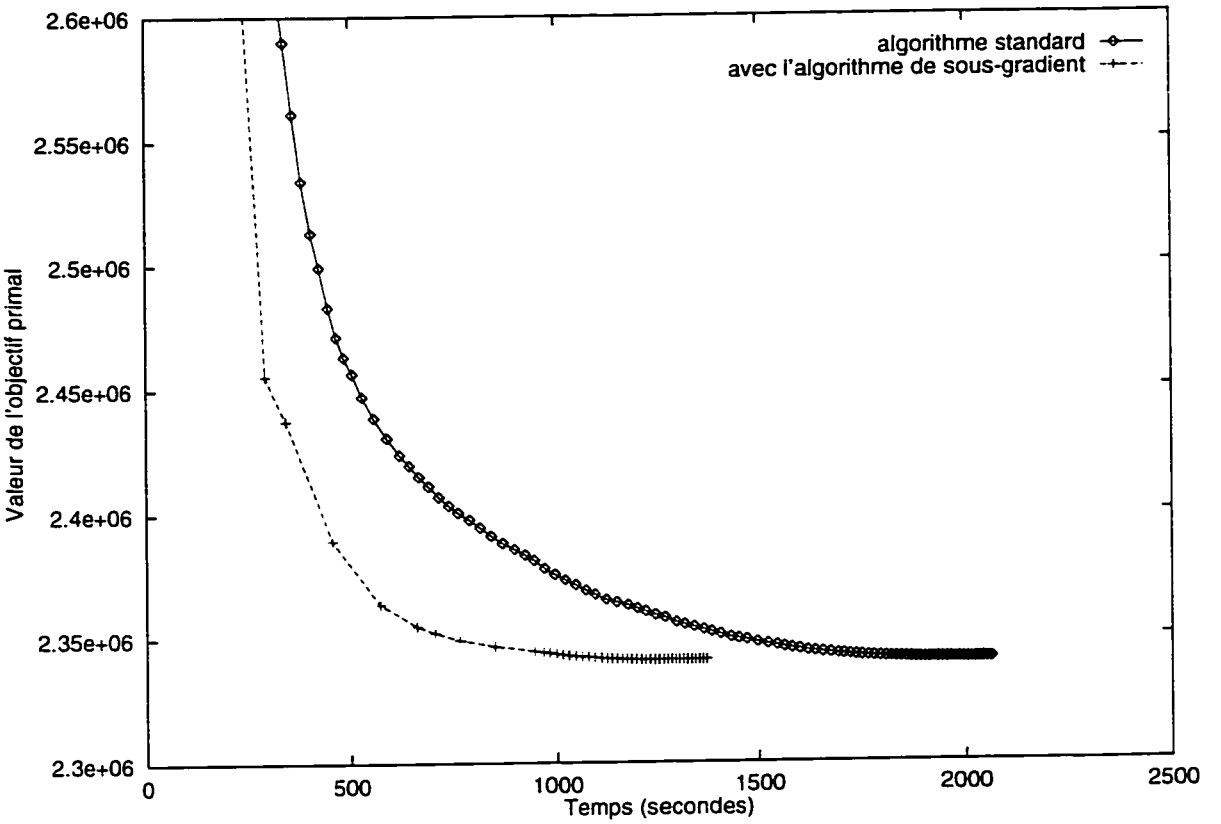


Figure 4.2: Comparaison de la valeur de la solution primale en fonction du temps, avec ou sans l'algorithme de sous-gradient

La figure 4.2 représente l'évolution de la valeur de la solution primale du problème maître restreint au cours du temps, pendant la résolution d'un problème de manière standard (trait plein) et à l'aide de l'algorithme de sous-gradient (trait pointillé). Dans cette figure, le temps en secondes est en abscisse, et la valeur de la solution primale en ordonnée.

Les points indiqués sur chacune des deux courbes correspondent aux itérations

pour lesquelles le problème maître restreint est résolu par l'algorithme du simplexe. Les itérations pour lesquelles le problème maître restreint est résolu par l'algorithme de sous-gradient ne donnent pas une nouvelle valeur à l'objectif primal et ne peuvent donc être représentées sur ce graphe.

Pour se repérer sur cette figure, il suffit de savoir que les itérations 8 à 32, pendant lesquelles les variables duales prennent des valeurs chaotiques, et où l'algorithme de sous-gradient semble être le plus efficace, correspondent à la période de 73 à 455 secondes pour l'algorithme de sous-gradient. En comparaison, pour la résolution standard la période de variables duales chaotiques correspond aux itérations 8 à 31 et aux temps de 72 à 560 secondes.

Il est clair d'après cette figure que l'utilisation de l'algorithme de sous-gradient accélère la résolution. Les deux courbes finissent avec des pentes similaires, ce qui suggère que l'influence de l'algorithme de sous-gradient est particulièrement utile au début de la résolution.

Le prochain chapitre décrit une autre façon d'améliorer les temps de résolution. Cette nouvelle méthode, qui utilise la perturbation, pourra se combiner avec la méthode de sous-gradient, en espérant que cela donne des résultats encore meilleurs.

Chapitre 5

Intégration d'un algorithme de stabilisation des variables duales à l'algorithme de génération de colonnes

Après l'algorithme de sous-gradient tel que décrit dans le chapitre précédent, le deuxième algorithme heuristique dual implanté et testé pour la résolution du problème maître dans un algorithme de génération de colonnes est un algorithme de contrôle des variables duales basé sur l'utilisation de variables de perturbation dans le problème primal. Cet algorithme est décrit dans la section 5.1; la section 5.2 explique son implantation dans un algorithme de génération de colonnes; enfin la section 5.3 présente des résultats numériques permettant d'évaluer l'efficacité d'une telle implantation, avec ou sans l'aide de l'algorithme développé au chapitre 4.

5.1 Les concepts théoriques

Cette section décrit l'algorithme de stabilisation des variables duales proposé par Du Merle, Villeneuve, Desrosiers et Hansen (1997). Cet algorithme s'applique à des problèmes linéaires résolus par une méthode de génération de colonnes. Considérons

le programme linéaire suivant:

$$\begin{array}{rrcl} & \min c^T X & & \\ (P) & \text{s.c. } AX & = & b \\ & X & \geq & 0 \end{array}$$

et le problème dual associé:

$$\begin{array}{rrcl} & \max b^T \pi & & \\ (D) & \text{s.c. } A^T \pi & \leq & c \end{array}$$

où X est le vecteur des variables de décision primales, c le vecteur des coûts de ces décisions, A la matrice des coefficients des variables de décisions dans les contraintes, b le vecteur des valeurs des membres droits des contraintes, et π le vecteur des variables duales.

Il est bien connu que la convergence d'une méthode de génération de colonnes est souvent lente lorsque P comporte un grand nombre de contraintes. Un moyen de stabiliser cette méthode est d'ajouter une **perturbation** aux contraintes de P, par le biais d'un vecteur de **variables de surplus** $y_- \geq 0$ et d'un vecteur de **variables d'écart** $y_+ \geq 0$. Ces variables sont bornées respectivement par les valeurs non négatives des vecteurs ϵ_- et ϵ_+. Elles ont pour but de perturber légèrement les contraintes primales, de manière à éviter la dégénérescence primale. En d'autres mots, le membre de gauche $A_i X$ d'une contrainte $A_i X = b_i, i \in I$ (où pour chaque $i \in I$, A_i représente la rangée i de la matrice A, et b_i l'élément i du vecteur b) peut être légèrement supérieur à son membre de droite b_i, en posant $A_i X - y_{i-} = b_i$, ou légèrement inférieur à celui-ci en posant $A_i X + y_{i+} = b_i$ (où pour chaque $i \in I$, y_{i-} représente l'élément i du vecteur y_-, et y_{i+} l'élément i du vecteur y_+). Ainsi, P se transforme en un nouveau problème:

$$\begin{array}{rrcl} & \min c^T X & & \\ & \text{s.c. } AX - y_- + y_+ & = & b \\ (P_\epsilon) & y_+ & \leq & \epsilon_+ \\ & y_- & \leq & \epsilon_- \\ & X, y_-, y_+ & \geq & 0. \end{array}$$

Un autre moyen de stabiliser la méthode de génération de colonnes est d'utiliser, comme en programmation non linéaire, la **méthode de pénalités exactes en norme l_1**, où la norme l_1 pour un vecteur X est définie par $\|X\|_{l_1} = \sum_{i \in I} |X_i|$. Dans cette méthode, la contrainte $AX = b$ est transférée dans l'objectif en lui associant une pénalité (constante) $\delta \geq 0$. Dans ce cas, le problème P se reformule comme suit:

$$(P_\delta) \quad \min_{X \geq 0} c^T X + \delta \|b - AX\|_{l_1} \;=\; \min_{X \geq 0} c^T X + \delta \sum_{i \in I} |b_i - A_i X|.$$

En utilisant les variables de surplus et d'écart $y_- \geq 0$ et $y_+ \geq 0$, et en posant $b_i - A_i X = y_{i+} - y_{i-}$, $\forall i \in I$, le problème P_δ se réécrit:

$$\begin{aligned} \min_{X \geq 0} c^T X + \delta \sum_{i \in I} |y_{i+} - y_{i-}| & \\ \text{s.c. } A_i X - y_{i-} + y_{i+} &= b_i, \ \forall i \in I \\ X, y_{i+}, y_{i-} &\geq 0, \ \forall i \in I. \end{aligned}$$

Mais $|y_{i+} - y_{i-}| = y_{i+} + y_{i-}$, car de par la nature de ces variables, pour chaque $i \in I$ une seule des deux variables y_{i+} et y_{i-} peut faire partie de la base, alors que l'autre vaut 0. Donc P_δ se réécrit:

$$\begin{aligned} \min_{X \geq 0} c^T X + \delta \sum_{i \in I} (y_{i+} + y_{i-}) & \\ \text{s.c. } A_i X - y_{i-} + y_{i+} &= b_i, \ \forall i \in I \\ X, y_{i+}, y_{i-} &\geq 0, \ \forall i \in I. \end{aligned}$$

Nous utiliserons par la suite des vecteurs δ_- et δ_+, au lieu d'une constante δ comme dans la méthode de pénalités exactes en norme l_1. Montrons que cette approche est équivalente. Définissons un nouveau problème:

$$\begin{aligned} & \min c^T X + \hat{\mu}^T (b - AX) \\ (\bar{P}) \qquad & \text{s.c. } AX = b \\ & X \geq 0 \end{aligned}$$

où $\hat{\mu}$ est un vecteur de multiplicateurs de dimension $|I|$. Puisque $b - AX = 0$ pour toute solution réalisable de $\bar{P}$, $\bar{P}$ est équivalent à P.

En appliquant la méthode de pénalités exactes en norme l_1 à $\bar{P}$, le problème $\bar{P}_\delta$, équivalent à P_δ, est obtenu comme suit (où $\hat{\mu}_i$ dénote l'élément i du vecteur $\hat{\mu}$):

$$\begin{aligned}(\bar{P}_\delta) \qquad & \min_{X\geq 0} c^T X + \hat{\mu}^T(b - AX) + \delta\|b - AX\|_{l_1} \\ = \; & \min_{X\geq 0} c^T X + \sum_{i\in I} \hat{\mu}_i(y_{i+} - y_{i-}) + \delta \sum_{i\in I}(y_{i+} + y_{i-}) \\ = \; & \min_{X\geq 0} c^T X + \sum_{i\in I}(\delta + \hat{\mu}_i)y_{i+} + \sum_{i\in I}(\delta - \hat{\mu}_i)y_{i-}.\end{aligned}$$

Ceci démontre que des vecteurs $\delta_+ = (\delta + \hat{\mu}_i | i \in I)$ et $\delta_- = (\delta - \hat{\mu}_i | i \in I)$ peuvent être utilisés à la place de la constante δ dans le problème P_δ.

La méthode de stabilisation qui sera utilisée ici combine les deux stratégies de P_ϵ et P_δ: le membre de gauche de la contrainte $A_i X = b_i, i \in I$ peut être inférieur ou supérieur au membre de droite correspondant, mais ce, à un certain coût δ_{i+} ou δ_{i-} et jamais par plus de ϵ_{i+} ou ϵ_{i-} respectivement. Cela donne le problème $(\tilde{P})$:

$$\min \; c^T\tilde{X} - \delta_- y_- + \delta_+ y_+ \tag{5.1}$$

$$\text{s.c. } A\tilde{X} - y_- + y_+ = b \tag{5.2}$$

$$y_- \leq \epsilon_- \tag{5.3}$$

$$y_+ \leq \epsilon_+ \tag{5.4}$$

$$\tilde{X}, y_-, y_+ \geq 0. \tag{5.5}$$

Le signe "moins" devant le vecteur de coefficients δ_- dans l'objectif entraînera une simplification de la notation par la suite. En associant les vecteurs de variables duales $\tilde{\pi}$, $-\omega_-$ et $-\omega_+$ (où $-\omega_- \leq 0$ et $-\omega_+ \leq 0$) aux contraintes (5.2), (5.3) et (5.4) de $\tilde{P}$, le problème dual $\tilde{D}$ associé à $\tilde{P}$ est donné par:

$$\max \; b^T\tilde{\pi} - \omega_-\epsilon_- - \omega_+\epsilon_+ \tag{5.6}$$

$$\text{s.c. } A^T\tilde{\pi} \leq c \tag{5.7}$$

$$-\tilde{\pi} - \omega_- \leq -\delta_- \quad (5.8)$$

$$\tilde{\pi} - \omega_+ \leq \delta_+ \quad (5.9)$$

$$\omega_-, \omega_+ \geq 0. \quad (5.10)$$

L'avantage de cette formulation est que le vecteur π des variables du problème dual perturbé $\tilde{D}$ est borné. En effet, les deux dernières contraintes (5.8) et (5.9) de $\tilde{D}$ se réécrivent $\delta_- - \omega_- \leq \tilde{\pi} \leq \delta_+ + \omega_+$. Les variables duales $\tilde{\pi}_i, i \in I$, qui ne sont pas dans l'intervalle $[\delta_-, \delta_+]$ sont pénalisées par un facteur ϵ_{i-} ou ϵ_{i+}. Il serait donc intéressant d'utiliser ces faits pour mieux contrôler les variables duales en cours de résolution et les faire converger plus rapidement.

Proposition: $\tilde{P}$ est équivalent à P (i.e. $y_-^* = y_+^* = 0$, où le symbole $*$ indique la valeur du vecteur à l'optimalité) si l'une des conditions suivantes est satisfaite:
(i) $\epsilon_- = \epsilon_+ = 0$,
(ii) $\delta_- < \tilde{\pi}^* < \delta_+$.

Preuve: Le résultat est évident lorsque la première condition $\epsilon_- = \epsilon_+ = 0$ est satisfaite. En effet, dans ce cas, les contraintes (5.3) et (5.4) de $\tilde{P}$ s'écrivent $y_- \leq 0$ et $y_+ \leq 0$. Jumelant ces contraintes aux contraintes de non négativité, les valeurs $y_-^* = 0$ et $y_+^* = 0$ sont obtenues.

Maintenant, supposons que la condition (ii) est satisfaite, i.e. $\delta_- < \tilde{\pi}^* < \delta_+$. Étant donné que le problème $\tilde{D}$ doit maximiser $b^T\tilde{\pi} - \omega_-\epsilon_- - \omega_+\epsilon_+$, et que $\epsilon-$, $\epsilon+$, ω_- et ω_+ doivent avoir des valeurs positives, il est facile de constater que $\omega_-^* = \omega_+^* = 0$. Or, les conditions de complémentarité associées aux contraintes (5.3) et (5.4) de $\tilde{D}$

$$(-\tilde{\pi}^* - \omega_-^* + \delta_-^*)y_-^* = 0$$

$$(\tilde{\pi}^* - \omega_+^* - \delta_+^*)y_+^* = 0,$$

se réécrivent:

$$(-\tilde{\pi}^* + \delta_-^*)y_-^* = 0$$

$$(\tilde{\pi}^* - \delta_+^*)y_+^* = 0.$$

Puisque $\delta_- < \tilde{\pi}^* < \delta_+$, les valeurs $y_-^* = 0$ et $y_+^* = 0$ sont obtenues. □

5.2 Intégration au logiciel GENCOL

L'algorithme de stabilisation des variables duales décrit à la section précédente s'implante facilement dans un algorithme de génération de colonnes. Il suffit de rajouter les variables de perturbation désirées au problème maître. À chaque itération de génération de colonnes, avant que le problème maître restreint ne soit résolu, il est possible de modifier les bornes et les coûts de ces nouvelles variables, de manière à contrôler les valeurs des variables duales. Nous avons effectué une telle implantation dans le logiciel GENCOL.

Pour cette implantation de l'algorithme de stabilisation, nous avons décidé, pour une itération donnée de la méthode de génération de colonnes, de borner les variables duales par un intervalle basé sur les valeurs des variables duales obtenues à l'itération précédente. De cette façon, nous disposons de valeurs intéressantes pour borner les variables duales. De plus, nous espérons atténuer le comportement chaotique des variables duales en début de résolution. Enfin, cette méthode assure à la fois une continuité et un dynamisme dans la variation des variables duales.

Un algorithme de perturbation existait déjà dans ce logiciel avant d'y implan-

ter l'algorithme de stabilisation des variables duales. Cet algorithme de perturbation ne permettait pas d'appliquer la théorie vue dans la section 5.1: les composantes des vecteurs δ_- et δ_+ y étaient toutes égales et, tout comme les vecteurs ϵ_- et ϵ_+, restaient fixes tout au long du processus de génération de colonnes. Cet algorithme de perturbation a été modifié pour que δ_- et δ_+ deviennent des vecteurs, à composantes possiblement différentes, et qu'ils puissent prendre de nouvelles valeurs à chaque itération de génération de colonnes. Les premiers tests modifiant les valeurs des vecteurs ϵ_- et ϵ_+ à chaque itération de génération de colonnes ont donné des temps de résolution trop longs, cette option n'a donc pas été conservée. Enfin, pour une solution duale initiale, la possibilité de lire individuellement les composantes de vecteurs δ_- et δ_+ initiaux a été mise en place, pour les cas où une approximation des valeurs des variables duales serait connue.

5.2.1 Les paramètres de l'algorithme de stabilisation des variables duales

Pour utiliser l'algorithme de stabilisation des variables duales tel qu'implanté dans le logiciel GENCOL, il faut tout d'abord que le paramètre booléen déterminant l'utilisation de la perturbation **SbbPerturbationUse** soit égal à 1, alors que sa valeur par défaut est 0. Il faut aussi que le paramètre booléen déterminant l'utilisation de l'algorithme de stabilisation des variables duales **SppPertAjUse** = 1 (c'est sa valeur par défaut).

De plus, l'ajustement des variables de perturbation peut n'avoir lieu qu'à partir d'une certaine itération de génération de colonnes, déterminée par le paramètre entier **SppPertAjItr**, qui par défaut vaut 10. Ce paramètre permet d'éviter l'ajustement

inutile des variables duales lors des premières itérations de génération de colonne, étant donné que les valeurs des variables duales sont souvent peu significatives dans ces itérations. Ceci est dû au fait que des variables artificielles de coût très grand sont utilisées. Comme l'algorithme de stabilisation des variables duales se base sur les valeurs des variables duales trouvées à l'itération précédente de génération de colonnes pour ajuster les coûts des variables de perturbation, il vaut mieux commencer à appliquer cet algorithme lorsque les variables duales commencent à prendre des valeurs plus raisonnables.

Pour démarrer l'algorithme de stabilisation des variables duales avec des vecteurs δ_- et δ_+ à composantes égales, il suffit d'utiliser les paramètres déjà implantés dans le logiciel GENCOL, soit les sept quadruplets **SppPertTaskStrong**, **SppPertSetPart**, **SppPertSetCov**, **SppPertFlow**, **SppPertGUB**, **SppPertCut** et **SppPertDef**. Chacun de ces paramètres permet de perturber un type particulier de contraintes. Un tel quadruplet donne, dans l'ordre, le coût des variables d'écart; une limite maximum aux variables d'écart; le coût des variables de surplus; et une limite maximum à ces variables. Pour chaque variable d'écart, la limite sera donnée par une valeur choisie au hasard entre 0 et la limite maximum donnée par le paramètre, selon une loi uniforme. Il en est de même pour les variables de surplus. Les valeurs ϵ_{i-} et $\epsilon_{i+}, i \in I$, utilisées dans l'algorithme de stabilisation découleront de ces valeurs.

Notons qu'il vaut mieux que les valeurs $\epsilon_{i-}, i \in I$ ne soient pas trop grandes, sinon tout le problème sera résolu en sous-recouvrant les tâches, solution qui est très loin de la solution recherchée. Cette remarque est aussi valable, à plus faible instance, pour les valeurs $\epsilon_{i+}, i \in I$ et le sur-recouvrement des tâches. Il faut malgré tout que les composantes ϵ_{i-} et $\epsilon_{i+}, i \in I$ soient différentes de 0 car, dans le cas contraire, les

colonnes de perturbation correspondantes ne seraient pas créées et l'algorithme de stabilisation des variables duales ne pourrait être utilisé.

Pour démarrer l'algorithme de stabilisation des variables duales avec des vecteurs δ_- et δ_+ de départ à composantes différentes, il faut fournir au logiciel GENCOL un fichier contenant une valeur approximative pour chaque variable duale. Le nom de ce fichier sera donné par le paramètre **SppPertAjDebut**. Les paramètres réels **SppPertAjCoefUnderDeb** (valant 1 ou plus, par défaut 1,2) et **SppPertAjCoefOverDeb** (valant entre 0 et 1, par défaut 0,8) seront des coefficients multiplicateurs pour calculer l'intervalle $[-\delta_{i+}, \delta_{i-}]$ bornant la valeur de la variable duale $\pi_i, i \in I$ à partir des valeurs données dans le fichier.

En dénotant par π_i^0 la i-ème valeur donnée par le fichier d'entrée, l'intervalle correspondant pour limiter la valeur de π_i est calculé ainsi:

$$\begin{aligned}
[-\delta_{i+}, \delta_{i-}] &= [\mathbf{Over} * \pi_i^0,\ \mathbf{Under} * \pi_i^0] && \text{si } \pi_i^0 \geq 1; \\
&= [-\mathbf{Under},\ \mathbf{Under}] && \text{si } -1 \leq \pi_i^0 < 1; \\
&= [\mathbf{Under} * \pi_i^0,\ \mathbf{Over} * \pi_i^0] && \text{si } \pi_i^0 < -1
\end{aligned}$$

(où pour plus de clarté les noms **SppPertAjCoefOverDeb** et **SppPertAjCoefUnderDeb** ont été remplacés simplement par "**Over**" et "**Under**").

Pendant la résolution, au début d'une itération g de génération de colonnes, les valeurs δ_{i-} et δ_{i+} se réajustent à l'aide des paramètres **SppPertAjCoefUnder** et **SppPertAjCoefOver**, valant par défaut 1,2 et 0,8 respectivement, de manière similaire à l'ajustement précédent:

$$\begin{aligned}
[-\delta_{i+}, \delta_{i-}] &= [\mathbf{Over} * \pi_i^{g-1},\ \mathbf{Under} * \pi_i^{g-1}] && \text{si } \pi_i^{g-1} \geq 1; \\
&= [-\mathbf{Under} * \pi_i^{g-1},\ \mathbf{Under} * \pi_i^{g-1}] && \text{si } -1 \leq \pi_i^{g-1} < 1; \\
&= [\mathbf{Under} * \pi_i^{g-1},\ \mathbf{Over} * \pi_i^{g-1}] && \text{si } \pi_i^{g-1} < -1
\end{aligned}$$

où encore une fois, pour plus de clarté, les noms **SppPertAjCoefOver** et **SppPertAjCoefUnder** ont été remplacés simplement par "**Over**" et "**Under**".

Tout au long de la résolution, les vecteurs δ_- et δ_+ devraient borner de mieux en mieux π. Idéalement, il arrive une itération de génération de colonnes pour laquelle $\delta_- < \pi < \delta_+$ et les variables de perturbation deviennent inactives. La résolution se continue alors normalement. Dans le cas contraire, une solution optimale de la relaxation linéaire ayant recours aux variables de perturbation est trouvée. Deux choix s'offrent alors pour poursuivre la rechercher d'une solution entière.

Premièrement, les variables de perturbation peuvent être toutes retirées et la résolution se poursuit alors normalement. Lorsque la solution perturbée est très mauvaise, le processus de résolution prend beaucoup de temps à récupérer et peut même revenir en phase I (avec des variables artificielles actives de coût très grand). Autrement, seulement quelques itérations supplémentaires sont nécessaires pour obtenir une solution optimale au problème non perturbé.

Deuxièmement, une borne inférieure peut être calculée à partir de la solution perturbée et les stratégies de branchement standard peuvent être invoquées. Dans ce cas, la borne inférieure peut tout simplement correspondre à la valeur primale de la solution perturbée, soit

$$b^T\tilde{\pi} - \omega_-\epsilon_- - \omega_+\epsilon_+.$$

Toutefois, la valeur $b^T\tilde{\pi}$ procure une meilleure borne valide. En effet, il est facile de vérifier que si $(\tilde{\pi}, \omega_-, \omega_+)$ est une solution réalisable pour le problème $\tilde{D}$, alors $\tilde{\pi}$ est aussi une solution réalisable pour le problème dual associé à P qui, rappelons-le, se formule

$$\begin{array}{lll} (D) & \max & b^T\pi \\ & \text{s.c.} & A^T\pi \leq c. \end{array}$$

De plus, comme $-\omega_-\epsilon_- - \omega_+\epsilon_+ \leq 0$, pour toute solution réalisable $(\tilde{\pi}, \omega_-, \omega_+)$ de $\tilde{D}$,

$$b^T\tilde{\pi} \geq b^T\tilde{\pi} - \omega_-\epsilon_- - \omega_+\epsilon_+.$$

5.3 Résultats

Une batterie de tests sur les paramètres de l'algorithme de stabilisation des variables duales ont permis de juger de l'efficacité de cet algorithme. Comme dans la section 4.3, ces tests ont porté sur la résolution de la relaxation linéaire de problèmes de type m-TSPTW tel que décrits au chapitre 3, et comportant 600 tâches. Les tests ont été effectués sur un ordinateur de type hp9000/715, à l'aide de la version 4.1b de GENCOL.

Les résultats de l'implantation de l'algorithme de stabilisation des variables duales dans le logiciel GENCOL sont mitigés, comme le montre le tableau 5.1. Ce tableau présente des valeurs obtenues pour un ensemble de 10 problèmes. Les trois premières colonnes indiquent les paramètres d'ajustement des variables duales utilisés pour la résolution des 10 problèmes. La première colonne ("Coefs de Début") indique les valeurs des paramètres **SppPertAjCoefOverDeb** et **SppPertAjCoefUnderDeb** s'ils sont actifs, c'est-à-dire dans le cas où un fichier de valeurs duales initiales est disponible. Le cas échéant, ces valeurs ont été trouvées par une résolution antérieure du même problème. La deuxième colonne ("Itr") donne la valeur du paramètre **SppPertAjItr** (indiquant l'itération à partir de laquelle l'algorithme de stabilisation des variables duales est appliqué). La troisième colonne ("Coefs") contient les valeurs des coefficients permettant de borner les variables duales par rapport aux valeurs trouvées à l'itération précédente, soit les valeurs des paramètres **SppPertAjCoefOver** et **SppPertAjCoefUnder**.

Les quatre colonnes suivantes du tableau indiquent, pour chacune des variations de ces paramètres, le meilleur et le pire temps de résolution (en secondes) pour les

10 problèmes considérés, ainsi que la moyenne et l'écart-type des temps de résolution obtenus pour la résolution de ces problèmes.

Quelles que soient les valeurs de ces paramètres, pour l'algorithme de stabilisation le temps de résolution moyen est meilleur sans l'utilisation de la perturbation. Même dans le meilleur cas, une détérioration du temps de résolution moyen de 2,4 % est observée par rapport au temps de résolution moyen obtenu avec la version standard.

Tableau 5.1: Temps de résolution selon les paramètres d'ajustement des coûts de perturbation

Coefs de Début	Itr	Coefs	min (s)	max (s)	moyenne (s)	écart-type (s)
sans stabilisation			**1265,0**	**2306,5**	**1722,4**	**259,8**
	10	(0,9; 1,1)	1476,1	2155,0	1764,2	187,7
(1,2; 1,6)	10	(0,9; 1,1)	1495,0	2127,9	1772,6	209,3
	20	(0,9; 1,1)	1442,1	2215,5	1798,2	216,8
	30	(0,8; 1,2)	1334,7	2316,6	1826,9	269,4
	30	(0,9; 1,1)	1356,4	2420,3	1840,2	274,1
	10	(0,9; 1,1)	1378,8	2358,3	1854,8	253,2
	20	(0,8; 1,2)	1354,0	2379,2	1861,1	264,2
(1,2; 1,6)	30	(0,8; 1,2)	1603,3	2357,4	1910,9	234,5

Des résultats légèrement meilleurs ont pu être obtenus dans le cas où une borne inférieure à la valeur de chaque variable duale était donnée dès le début. Une telle borne peut être calculée par le "détour" minimum d'une tâche: c'est le coût minimum à payer pour qu'un chemin couvre la tâche en question. La valeur $d(k)$ du détour pour la tâche k est donné par

$$d(k) = \min_{(i,k),(k,j),(i,j)\in A} \{c_{ik} + c_{kj} - c_{ij}\}.$$

Malheureusement, les valeurs des détours sont longues à calculer et, même quand elles sont disponibles, les meilleurs améliorations aux temps de résolution sont seulement de l'ordre de 10 %.

Il est intéressant de noter que pour les ajustements de paramètres donnant les temps de résolution moyens les plus proches du temps de résolution standard (c'est-à-dire les meilleurs paramètres pour l'algorithme de stabilisation), l'écart-type des temps de résolution est un peu plus petit avec l'algorithme de stabilisation qu'avec l'algorithme résolution standard. De bons ajustements des paramètres de l'algorithme de stabilisation permettent donc des temps de résolution plus homogènes.

L'algorithme de stabilisation et l'algorithme de sous-gradient développé au chapitre 4 n'agissent pas sur les mêmes composantes du problème maître restreint et peuvent être utilisés simultanément sans conflit. Cette combinaison d'algorithmes a donc elle aussi été testée. De manière un peu surprenante, l'utilisation simultanée de l'algorithme de stabilisation et de l'algorithme de sous-gradient peut donner de meilleurs résultats que lorsque chacun de ces algorithmes est utilisé seul. Les résultats des tests de cette combinaison d'algorithmes sont présentés dans le tableau 5.2. Pour ces tests, nous avons utilisé les valeurs des paramètres qui correspondaient aux deux meilleurs résultats en terme de temps moyen de résolution, pour chacun des algorithmes. Ces valeurs ont été trouvées à la section 4.3 pour l'algorithme de sous-gradient et à la présente section pour l'algorithme de stabilisation des variables duales. Les temps de résolution pour les quatre combinaisons possibles de ces ensembles de paramètres ont été évalués. La première colonne du tableau 5.2 indique l'ensemble de paramètres utilisé pour l'algorithme de sous-gradient: "meilleur" si le meilleur ensemble de paramètres est utilisé, "2ème" si c'est le deuxième meilleur ensemble de paramètres qui est utilisé, "non" si l'algorithme de sous-gradient n'est pas utilisé. De même, la deuxième colonne indique l'ensemble de paramètres utilisé pour l'algorithme de stabilisation des variables duales. Enfin, tel que pour le tableau 5.1, les quatre dernières colonnes indiquent les temps de résolution le plus rapide et le plus lent, la moyenne

et l'écart-type, en secondes, obtenus avec chaque combinaison de paramètres pour un ensemble de 10 problèmes.

Tableau 5.2: Temps de résolution selon les paramètres d'ajustement des coûts de perturbation et les paramètres de l'algorithme de sous-gradient combinés

sous-gradient	stabilisation	min (s)	max (s)	moyenne (s)	écart-type (s)
meilleur	2ème	1053,9	1516,2	1210,5	121,4
meilleur	**non**	1109,6	1423,6	1240,0	99,8
2ème	2ème	1112,5	1545,4	1240,6	119,6
2ème	non	1081,0	1480,8	1260,0	123,3
meilleur	meilleur	1115,9	1851,9	1344,3	196,2
2ème	meilleur	1173,6	1804,7	1403,5	175,1
non	**non**	**1265,0**	**2306,5**	**1722,4**	**259,8**
non	**meilleur**	1476,1	2155,0	1764,2	187,7
non	2ème	1495,0	2127,9	1772,6	209,3

Le tableau 5.2 indique que les meilleurs résultats ont été obtenus en combinant les meilleurs paramètres pour l'algorithme de sous-gradient avec la deuxième meilleure série de paramètres pour l'algorithme de stabilisation des variables duales. Ces paramètres correspondent tous aux paramètres par défaut. Ce résultat correspond à une amélioration totale de 29,7 % du temps de résolution moyen de la relaxation linéaire du problème maître, comparativement à une amélioration de 28 % lorsque l'algorithme de sous-gradient est utilisée sans l'algorithme de stabilisation. La différence n'est pas frappante, mais elle est tout de même étonnante. Puisque l'algorithme de sous-gradient accélère la résolution, mais que l'algorithme de stabilisation la ralentit, l'effet de la combinaison des deux algorithmes aurait pu être une valeur intermédiaire entre les deux résultats.

Comme lorsque chacun des algorithmes est utilisé seul avec des paramètres assez bons, la combinaison des deux algorithmes donne un l'écart-type des temps de résolution plus petit que pour la résolution standard. Les temps de résolution sont

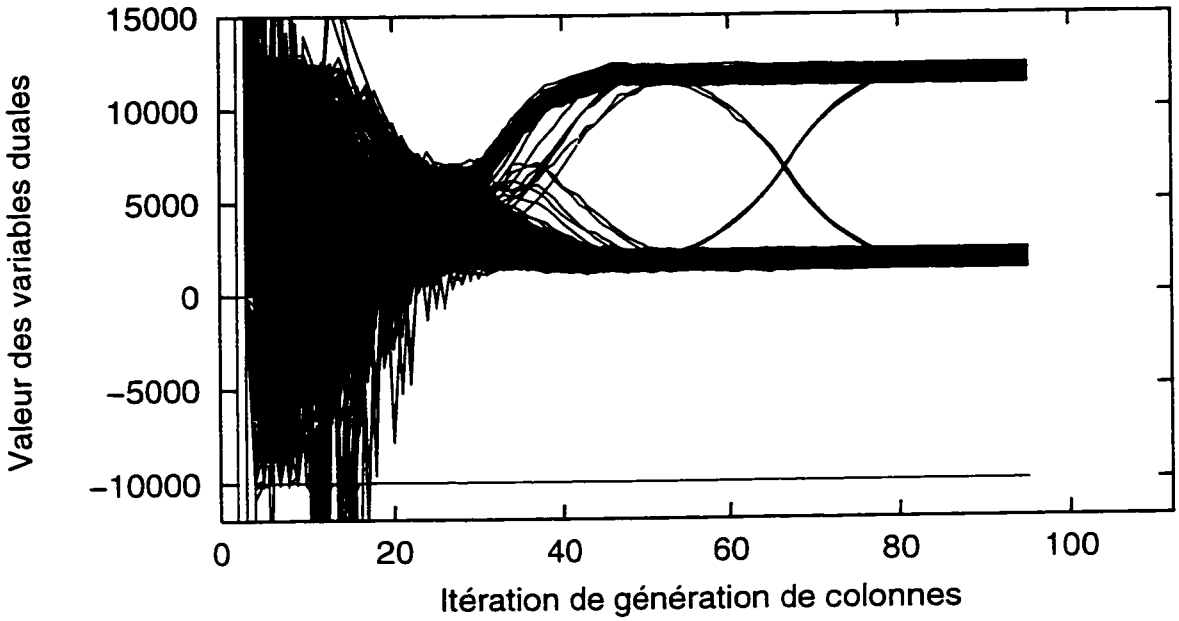


Figure 5.1: Comportement des variables duales avec l'algorithme de stabilisation

donc plus homogènes lorsque les deux algorithmes sont utilisés ensemble que lorsqu'aucun de ces deux algorithmes n'est utilisé. Par contre, il est difficile de dire si l'utilisation combinée des deux algorithmes donne un meilleur écart-type que l'utilisation d'un seul de ces algorithme.

La figure 5.1 permet d'apprécier l'influence de l'algorithme de stabilisation sur l'évolution des valeurs des variables duales. Cette figure représente les valeurs des variables duales en fonction de l'itération, à la même échelle que les figures 1.1 et 4.1. Dans cette résolution, la perturbation est appliquée de manière classique (c'est-à-dire sans ajustement des coûts) dès la première itération. Cette première perturbation est très grossière (elle limite les variables duales entre -10000 et 10000) et n'existe que dans le but d'obliger le logiciel GENCOL à créer des variables de perturbation et à les ajouter au problème (autrement elles ne seraient pas créées). Des variables artificielles sont actives pendant les trois premières itérations, ce qui rend les valeurs des

variables duales très grandes. L'ajustement des coûts des variables de perturbation ne commence à se faire qu'à l'itération 10. Le nombre de variables de perturbation actives varie tout au long de la résolution. Entre les itérations 10 et 20, il est de l'ordre de 400 à 500 (pour 609 variables duales). Puis, il baisse doucement jusqu'à l'itération 69, à partir de laquelle plus aucune variable de perturbation n'est utilisée. L'analyse du nombre de variables de perturbation utilisées (i.e., de valeur non nulle) par rapport au graphe des variables duales indique qu'entre les itérations 10 et 20, la perturbation est très utilisée mais peu influente sur l'évolution des variables duales. Apparemment, d'autres décisions sont plus intéressantes pour améliorer la solution que d'arrêter d'utiliser des variables de perturbation. L'action des variables de perturbation commence à se faire sentir à l'itération 20, et jusque vers l'itération 45 les coûts des variables de perturbation guident les variables duales vers des valeurs jugées optimales. Mais au lieu d'encadrer les variables duales vers la bonne valeur, les coûts des variables de perturbation ont plutôt l'effet contraire de retenir les variables duales vers leur valeur précédente (valeur sur laquelle ces coûts sont basés), et cela ralentit la résolution. Vers l'itération 45, il reste déjà moins de 100 variables de perturbation actives et elles n'ont que peu d'influence sur le contrôle des variables duales. C'est pourtant à ce moment que leur action doit être la plus utile, car les bornes qu'elles définissent sont enfin bonnes. Pour les variables duales qui convergent vers de petites valeurs (entre 1000 et 3000), la largeur de l'intervalle défini par les coûts des variables de perturbation est de 200 à 600. Pour les variables duales qui comprennent un coût fixe et qui convergent vers des valeurs élevées (entre 10500 et 12000), cette largeur est de 2100 à 2500 environ. La manière dont l'algorithme est appliqué fait que les variables duales les plus petites, qui dans notre cas sont aussi les plus nombreuses, sont bornées plus finement.

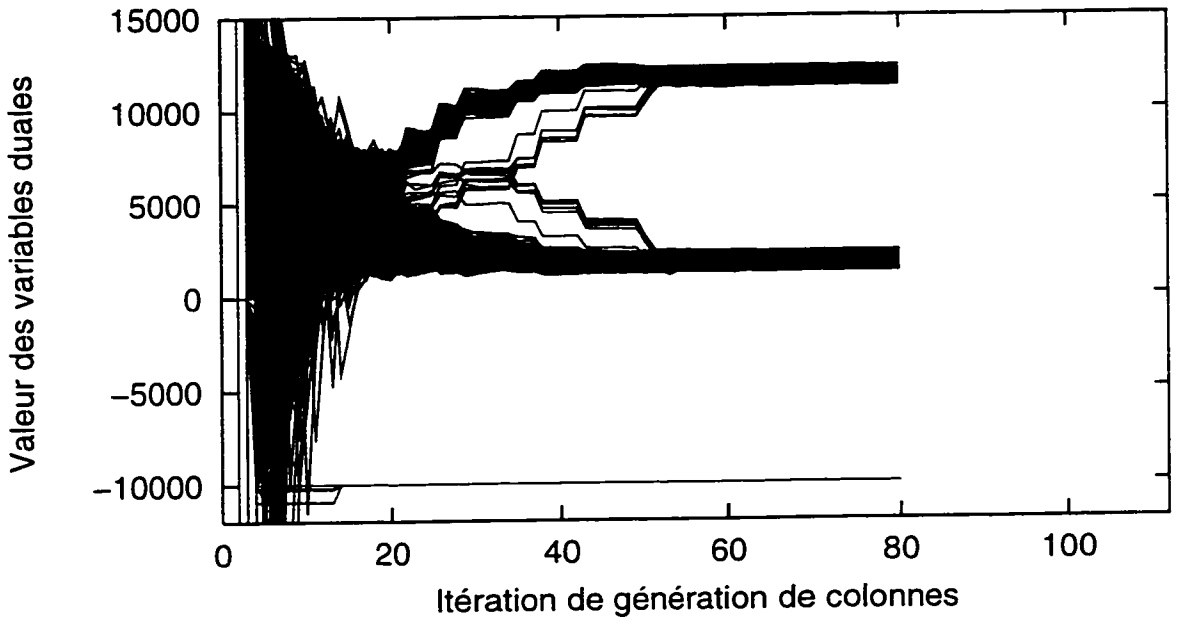


Figure 5.2: Comportement des variables duales avec l'algorithme de stabilisation et l'algorithme de sous-gradient combinés

La figure 5.2 représente l'évolution des valeurs des variables duales, lorsque les deux algorithmes implantés sont utilisés, soit l'algorithme de sous-gradient et l'algorithme de stabilisation des variables duales. Cette figure représente les valeurs des variables duales en fonction de l'itération, pour le même problème qu'aux figures 1.1, 4.1 et 5.1 et à la même échelle (valeurs des variables duales en fonction de l'itération de génération de colonnes).

L'évolution des variables duales au cours de cette résolution se fait comme suit. Entre les itérations 1 et 3 des variables artificielles sont actives, ce qui rend les variables duales très grandes. L'algorithme de sous-gradient est utilisé entre les itérations 4 et 49 pour environ quatre problèmes maîtres restreints sur cinq de ces itérations. Par la suite, seul l'algorithme du simplexe est utilisé pour résoudre le problème maître restreint car le gap entre la solution duale et la borne supérieure, donnée par la

dernière solution primale trouvée, est toujours plus petit que le gap permis par le paramètre **SgMinGap**.

Comme dans le cas où l'algorithme de stabilisation est utilisé seul, les variables de perturbation sont très utilisées au début, puis leur utilisation décroît tout au long de la résolution. Dans l'exemple présent, les coûts des variables de perturbation sont ajustés selon l'algorithme de stabilisation dès le début de la résolution. Entre les itérations 10 et 20 environ, de 300 à 400 variables de perturbation sont utilisées. Ce nombre baisse en-dessous de 100 à partir de l'itération 30 et, dès l'itération 52, plus aucune variable de perturbation n'est active.

Des caractéristiques des figures 1.1, 4.1 et 5.1 se retrouvent dans la figure 5.2. Les variables duales restent chaotiques pour les premières itérations, soient les itérations 1 à 22 dans ce cas-ci. Comme c'était le cas lorsque l'algorithme de stabilisation des variables duales est utilisé seul, les variables duales restent pendant plus d'itérations à des valeurs moyennes avant de se séparer et de converger vers leur valeur finale. Ceci s'observe particulièrement pendant les itérations 16 à 22, et continue jusqu'à l'itération 52 où plus aucune variable de perturbation n'est active.

D'autre part, ce graphe arbore les soubresauts déjà observés lorsque l'algorithme de sous-gradient est utilisé seul. Ces soubresauts correspondent à l'ajustement plus précis et plus brutal des variables duales par l'algorithme du simplexe suite à plusieurs itérations pendant lesquelles le problème maître restreint a été résolu par l'algorithme de sous-gradient.

L'évolution des variables duales lorsque les deux algorithmes de sous-gradient et de stabilisation des variables duales sont utilisés comporte les caractéristiques des deux

algorithmes. L'accélération promise par l'algorithme de sous-gradient pourrait être ralentie par l'algorithme de stabilisation des variables duales, qui a plutôt tendance, tel qu'il est appliqué, à empêcher les variables duales de converger vers leurs valeurs optimales. Au contraire, en moyenne, le temps d'exécution est légèrement accéléré par rapport au temps de résolution du même problème lorsque l'algorithme de sous-gradient est utilisé seul. Paradoxalement, les variables duales dans la résolution utilisant les deux méthodes convergent beaucoup plus lentement, en fonction du temps, que lorsque la méthode de sous-gradient est utilisée seule.

La figure 5.3 représente l'évolution de l'objectif primal par rapport au temps en secondes, pour les quatre algorithmes de résolution considérés dans ce mémoire. La résolution standard y est représentée par un trait plein, et les autres résolutions par différents types de traits pointillés, avec des 'x' pour l'algorithme de sous-gradient, des '+' pour l'algorithme de stabilisation, et des '□' pour la résolution à l'aide des deux algorithmes combinés de stabilisation et de sous-gradient. Dans cet exemple, la résolution à l'aide de l'algorithme de stabilisation a été un peu plus rapide que la résolution standard, tandis que la résolution utilisant l'algorithme de sous-gradient fut bien plus rapide, et celle utilisant les deux algorithmes de stabilisation et de sous-gradient, encore plus rapide. Dans les cas où l'algorithme de stabilisation est utilisé, les coûts des variables de perturbation changent tout au long de la résolution, ce qui explique le fait que la valeur de l'objectif puisse augmenter. Comme ces coûts, qui bornent les variables de perturbation, s'ajustent tout au long de la résolution, à la fin de la résolution les variables de perturbation sont toutes à l'intérieur de leurs bornes respectives. Dans ce cas aucune variable de perturbation n'est utilisée donc le problème n'a pas besoin d'être déperturbé.

Pour l'algorithme de stabilisation, la valeur de l'objectif primal varie longtemps autour de la valeur optimale avant de l'atteindre. Dans le cas où les deux algorithmes de stabilisation et de sous-gradient sont utilisés conjointement, l'objectif rejoint très vite une valeur presqu'optimale et prend ensuite un peu de temps à ajuster cette valeur.

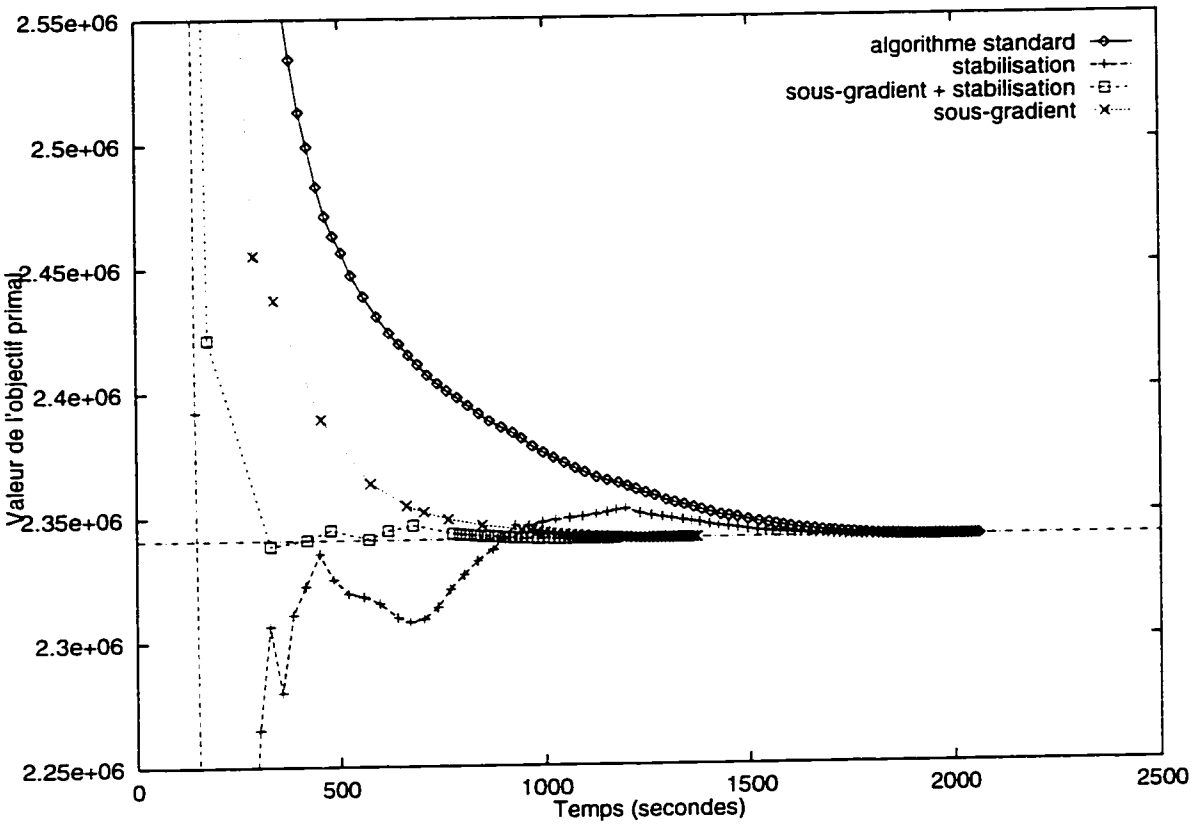


Figure 5.3: Rapidité de convergence avec l'algorithme de stabilisation et l'algorithme de sous-gradient combinés par rapport à l'algorithme standard

La comparaison de la convergence des variables duales par rapport au temps de résolution pour l'algorithme de stabilisation utilisé seul (figure 5.4), comparativement à l'utilisation conjointe des deux algorithmes (figure 5.5), indique clairement l'avantage de combiner les deux algorithmes en terme de temps. La raison pour laquelle cette combinaison d'algorithmes est avantageuse sur chacun des algorithmes

utilisés seuls n'est pas facile à identifier. Une explication possible est que l'algo-

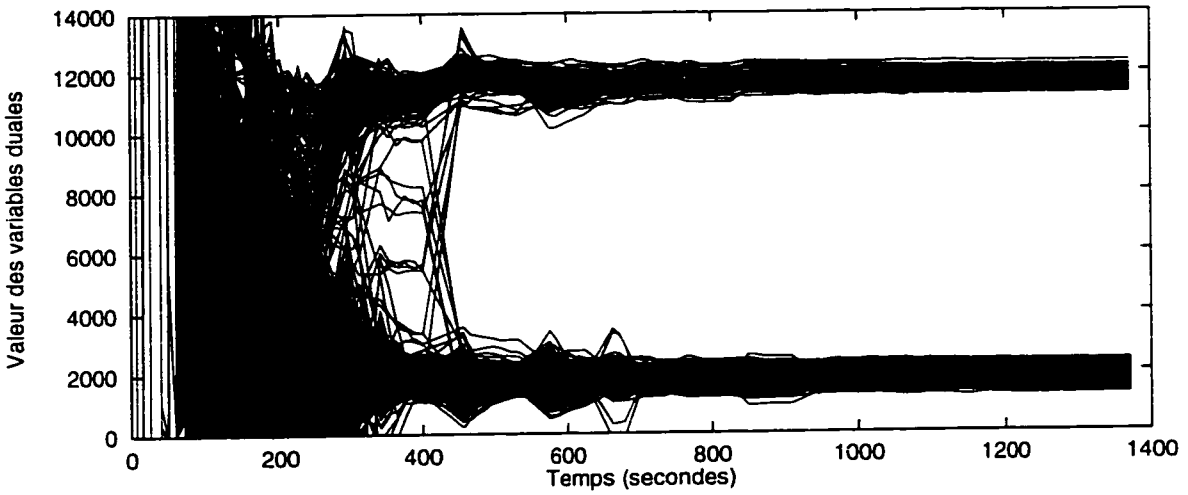


Figure 5.4: Comportement des variables duales avec l'algorithme de stabilisation, par rapport au temps

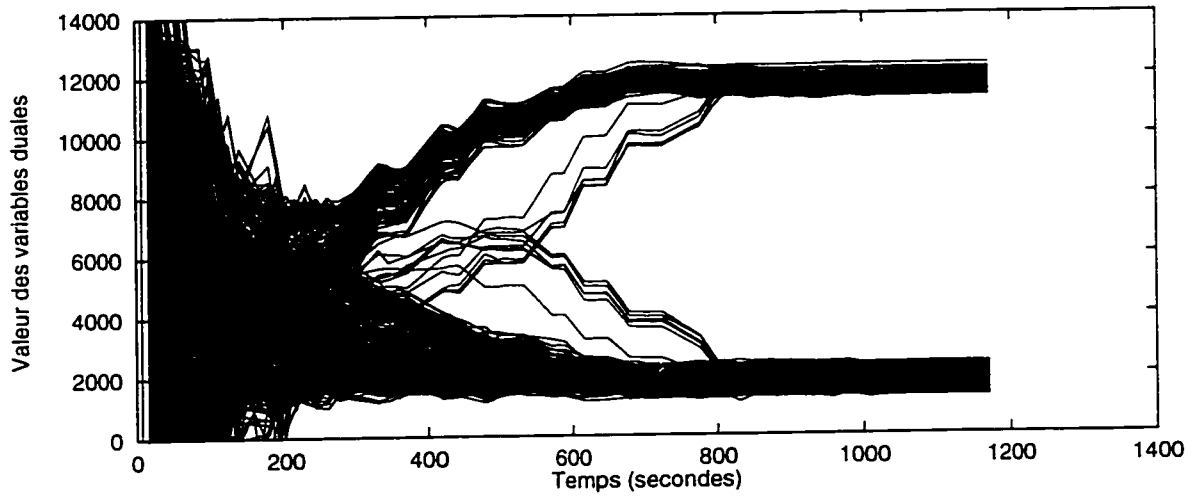


Figure 5.5: Comportement des variables duales avec l'algorithme de stabilisation et l'algorithme de sous-gradient combinés, par rapport au temps

rithme de sous-gradient permet de réduire considérablement le temps de résolution du problème lorsque celui-ci est résolu à l'aide de l'algorithme de stabilisation. Tout le temps qui était perdu à optimiser des problèmes maîtres restreints, devenus très lourds par l'ajout de nombreuses variables de perturbation, est maintenant sauvé

grâce à l'algorithme de sous-gradient, ce qui permet à l'algorithme de perturbation d'avoir lui-même une certaine utilité. Mais d'autres analyses doivent être faites afin d'expliquer le comportement de ces deux algorithmes.

La théorie derrière l'algorithme de stabilisation des variables duales reste intéressante et la recherche se poursuit afin d'obtenir de meilleurs résultats à partir de cet algorithme ou d'algorithmes similaires.

Conclusion

Dans ce mémoire de maîtrise, nous avons vu l'implantation de deux algorithmes heuristiques pour la résolution du problème maître dans un contexte de génération de colonnes.

Le premier algorithme heuristique, l'algorithme de sous-gradient, permet de résoudre le problème maître restreint de manière beaucoup plus rapide que l'algorithme du simplexe, mais non optimale. Cette résolution duale fournit une bonne approximation des variables duales qui sont transférés au sous-problème. Régulièrement, la résolution du problème maître restreint est faite jusqu'à l'optimalité par l'algorithme du simplexe. Cette alternance d'algorithmes heuristique et optimal permet de perdre moins de temps à calculer l'optimalité de problèmes intermédiaires moins significatifs, ce qui en définitive se traduit par une résolution 28% plus rapide en moyenne.

Le deuxième algorithme heuristique, l'algorithme de stabilisation des variables duales, permet de stabiliser les variables duales au cours des itérations de génération de colonne, en utilisant pour cela des variables de perturbation. En effet, les coûts de variables de perturbation peuvent borner les variables duales. En ajustant ces coûts selon les valeurs des variables duales d'une itération précédente, les variables duales deviennent bornées et oscillent moins que dans le processus standard de résolution, convergeant donc plus rapidement vers leurs valeurs optimales. Malheureusement, malgré que les variables duales convergent un peu plus rapidement, en terme d'itérations, le

problème maître restreint, alourdi par toutes les variables de perturbation, prend beaucoup plus de temps à résoudre et en définitive la résolution est en moyenne toujours plus lente que dans l'algorithme standard sans perturbation. Néanmoins, la recherche se poursuit dans cette direction.

Ces deux algorithmes donnent des temps de résolution plus homogènes, qu'ils soient utilisés seuls ou combinés ensemble. D'un autre côté, la combinaison de ces deux algorithmes crée une synergie qui rend la résolution jusqu'à 29,7% plus rapide que l'algorithme standard en moyenne (ce qui est 2,4% plus rapide que l'algorithme de sous-gradient seul). En particulier il est très intéressant de constater que cette combinaison fait converger le problème plus rapidement vers la solution optimale.

Ces deux algorithmes marquent le début au GERAD de la recherche pour l'accélération de la résolution du problème maître restreint et, espérons-le, ouvrent la voie à de nombreuses nouvelles méthodes toutes plus efficaces, rapides et optimales les unes que les autres.

Bibliographie

[1] BALAS E. et HO, A. (1980). Set Covering Algorithms Using Cutting Planes, Heuristics and Subgradient Optimization: A Computational Study. *Mathematical Programming* Vol. 12, 37–60.

[2] BEASLEY, J.E. (1992). *Lagrangean Relaxation.* The Management School, Imperial College, Londres, Royaume-Uni.

[3] CAPRARA, A., FISHETTI, M. et TOTH, P. (1996). A Heuristic Algorithm for the Set Covering Problem. Document de travail.

[4] CHVATÀL, V. (1979). A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research* Vol. 4, No. 3, 233–235.

[5] DANTZIG, G.B. et WOLFE, P. (1961). Decomposition Principle for Linear Programs. *Operations Research* Vol. 8, 101–111.

[6] DESROSIERS, J., DUMAS, Y., SOLOMON, M.M., et SOUMIS, F. (1995). *Time Constrained Routing and Scheduling.* In M.O. Ball *et al.* (eds.), *Network Routing*, Handbooks in Operations Research and Management Science 8. Elsevier Science, Amsterdam, 35–139.

[7] DESROCHERS, M., DESROSIERS, J., et SOLOMON, M. (1992). A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research* Vol. 40, 342–354.

[8] DESROSIERS, J., SOUMIS, F., et DESROCHERS, M. (1984). Routing with Time Windows by Column Generation. *Networks* Vol. 14, 545–565.

[9] DESROSIERS, J., SAUVÉ, M. et SOUMIS, F. (1988). Lagrangian Relaxation Methods for Solving the Minimum Fleet Size Multiple Traveling Salesman Problem with Time Windows. *Management Science* Vol. 34, 1005–1022.

[10] DESROSIERS, J., DUMAS, Y., SOLOMON, M.M., et SOUMIS, F. (1992). *Time Constrained Routing and Scheduling.*

[11] DU MERLE, O. (1995) *Interior Point and Cutting Plane Method: a New Algorithm for Convex Optimization and Large Scale Structured Linear Programming.* Thèse de Doctorat, Université de Genève, Hautes Études Commerciales, Suisse.

[12] DU MERLE, O., VILLENEUVE, D., DESROSIERS, J. et HANSEN, P. (1997). *Stabilisation dans le cadre de la génération de colonnes.* Cahiers du GERAD G-97-08, École des HEC, Montréal, Canada.

[13] FISHER, M.L. (1981). The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science* Vol. 27, No 1, 1–18.

[14] FISHER, M.L. (1985). An Application Oriented Guide to Lagrangean Relaxation. *Interfaces* Vol. 15, 10–21.

[15] FISHER, M.L., JAIKUMAR, R. et VAN WASSENHOVE, L. (1986). A Multiplier Adjustment Method for the Generalized Assignment Problem. *Management Science* Vol. 32, No 9, 1095–1103.

[16] FISHER, M.L. et KEDIA, P. (1990). Optimal Solution of Set Covering/Partitioning Problems Using Dual Heuristics. *Management Science* Vol. 36, No 6, 674–688.

[17] GEOFFRION, A.M. (1974). Lagrangian Relaxation and Its Uses in Linear Programming. *Mathematical Programming Study* Vol. 2, 82-114.

[18] GOFFIN, J.-L. (1977). On the Convergence Rate of Subgradient Optimization. *Mathematical Programming* Vol. 13, 329–347.

[19] GOFFIN, J.-L., HAURIE, A. et VIAL, J.-P. (1992). Decomposition and Nondifferentiable Optimization with the Projective Algorithm. *Management Science* Vol. 38, No. 2, 284–302.

[20] GUIGNARD, M. et SPIELBERG, K. (1979a). A Direct Dual Method for the Mixed Plant Location Problem with Some Side Constraints. *Mathematical Programming*, Vol. 17, 198–228.

[21] GUIGNARD, M. et SPIELBERG, K. (1979b). *A Direct Dual Approach to Transshipment Formulation for Multi-Layer Network Problems with Fixed Charges.* Rapport Technique #43, Department of Statistics, University of Pennsylvania, États-Unis.

[22] HELD, M., WOLFE, P. et CROWDER, H.D. (1974). Validation of Subgradient Optimization. *Mathematical Programming* Vol. 6, 62–88.

[23] KEDIA, P. (1985). *Lagrangian Multiplier Adjustment Method for Solving Certain Combinatorial Problems.* Thèse de Doctorat, University of Pennsylvania, États-Unis.

[24] KOHL, N. (1994). *An Improvement of the Subgradient Method.* Rapport Technique IMM-REP-1994-20, Institute of Mathematical Modelling, The Technical University of Denmark, DK-2800 Lyngby, Danemark.

[25] KOHL, N. (1995). *Exact Methods for Time Constrained Routing and Related Scheduling Problems*. Thèse de Doctorat, The Technical University of Denmark, DK-2800 Lyngby, Danemark.

[26] KOHL, N. et MADSEN, O.B.G. (1997). An Optimization Algorithm for the Vehicle Routing Problem with Time Windows based on Lagrangean Relaxation. *Operations Research* Vol. 45, 395–406.

[27] KOLEN, A.W.J., RINNOOY KAN, A.H.G., et TRIENEKENS, H.W.J.M. (1987). Vehicle Routing with Time Windows. *Operations Research* Vol. 35, 266–273.

[28] LAVIGNE, J. (1996). *Le problème de tournées de véhicules avec fenêtres de temps et dépôts multiples*. Mémoire de Maîtrise, École Polytechnique, Montréal, Canada.

[29] LEMARÉCHAL, C. (1989). Non Differentiable Optimization. *Handbooks of Operations Research and Management Science, volume 1: Optimization*, ed. G.L. Nemhauser, A.G.H. Rinnooy Kan and M.J. Todol, North-Holland, Amsterdam, 529–572.

[30] MAHEY, P. (1982). Decomposition of Large-Scale Linear Programs by Subgradient Optimization. *Matemática Aplicada e Computacional* Vol. 1, No 2, 121–134.

[31] MAHEY, P. (1986). Méthodes de décomposition et décentralisation en programmation linéaire. *R.A.I.R.O. Recherche Opérationnelle* Vol. 20, No 4, 287–306.

[32] MARSTEN, R.E. (1974). An Algorithm for Large Set Partitioning Problems. *Management Science* Vol. 20, 770–787.

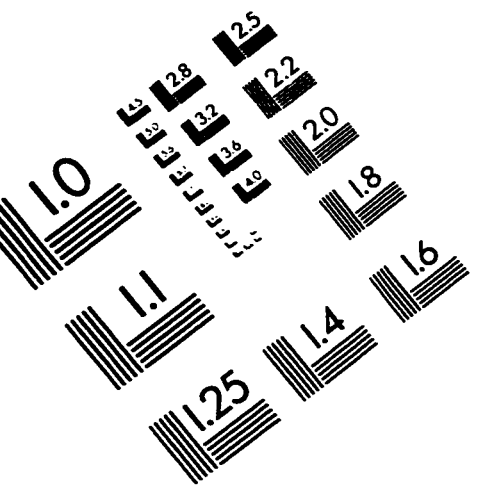
[33] NEMHAUSER, G.L. et WOLSEY, R.A. (1988). *Integer and Combinatorial Optimization.* Wiley-Interscience, New York, États-Unis.

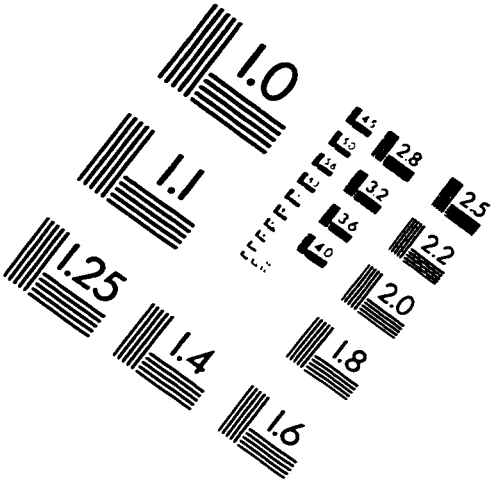
[34] SOLOMON, M.M. (1987) Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints. *Operations Research* Vol. 35, 254–265.

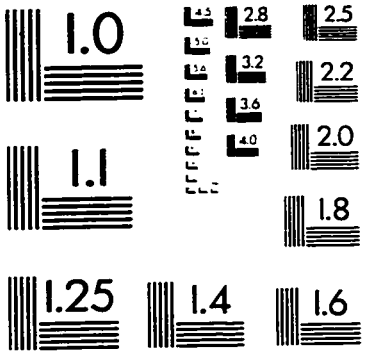
[35] SOLOMON, M.M. et DESROSIERS, J. (1988). Time Window Constrained Routing and Scheduling Problems. *Transportation Science* Vol. 22, 1–13.

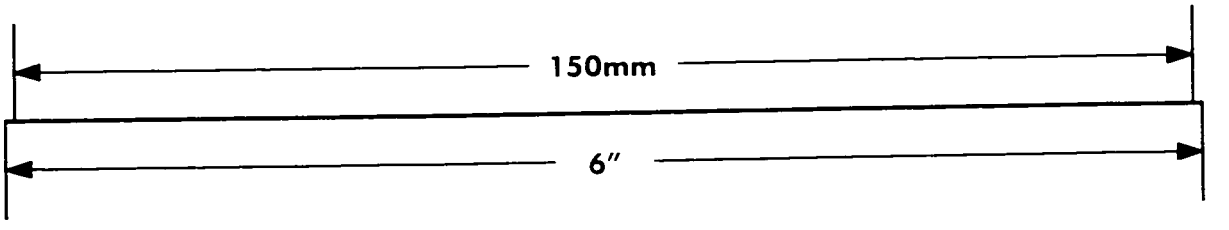
[36] ZOWE, J. (1985). Nondifferentiable Optimization. *Computational Mathematical Programming* Vol. F15, ed. Schittkowski, K., NATO ASI Series, Springer-Verlag, Berlin, Allemagne.

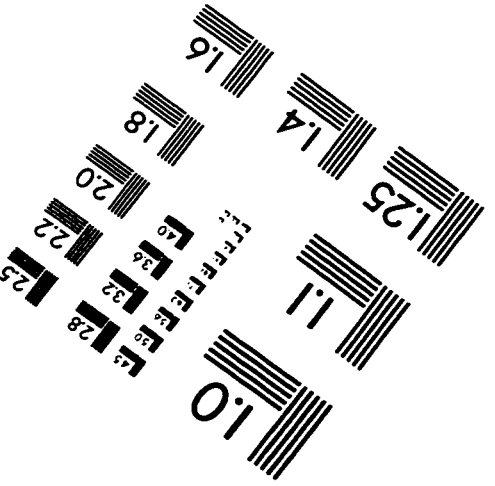
IMAGE EVALUATION
TEST TARGET (QA-3)

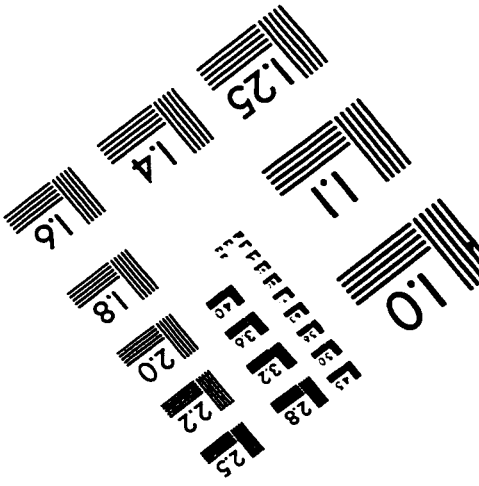












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989